AFIT/GST/ENS/93M-10

A FUZZY LOGIC OPTIMAL CONTROL LAW SOLUTION TO THE CMMCA TRACKING PROBLEM

THESIS

Randy K. Nelson, Major, USAF

AFIT/GST/ENS/93M-10

AFIT/GST/ENS/93M-10

A FUZZY LOGIC OPTIMAL CONTROL LAW SOLUTION TO THE CMMCA TRACKING PROBLEM

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Randy E. Nelson, B.S.
Major, USAF

March, 1993

THESIS APPROVAL

STUDENT: Major Randy E. Nelson CLASS: GST 93-M

THESIS TITLE: A Fuzzy Logic Optimal Control Law Solution to the CMMCA Tracking Problem

DEFENSE DATE: 5 MAR 93

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Advisor:	Dr Donald J. Caughlin/EN	
Reader:	Colonel Thomas F. Schuppe/LA	

Preface

The purpose of this thesis was to develop a set of control laws, implemented in a simulation, to allow the Cruise Missile Control Aircraft (CMMCA) to radar track cruise missiles and achieve 100% radar coverage during test flights. The CMMCA was forced to perform a series of intercepts on set points recomputed every ten seconds. Fuzzy logic formed a vital function in determining where the set point would be, and used current and future missile position data.

This thesis development and completion was only possible through the tireless help and motivation provide by my advisor, Col Don Caughlin. His "optimal guidance" prevented many a random flight path in my work on this thesis for so many months. I also want to extend my gratitude to Col Tom Schuppe for suggesting this problem, pointing me toward Col Caughlin, and for his assistance as my reader. I thank both members of my committee for their superhuman restraint in not equating the fuzzy logic used in the thesis to any of my thinking processes.

Thanks to all my fellow GST's and GOR's for their concern in my progress, and to Capt Scott Goehring for carrying me in CM II while I worked frantically on finishing my thesis.

A very special thanks to my wonderful wife Mary. She has been the rock wherein I've found my sanity during the last six months. Her strength has re-energized me when I thought I was hopelessly bogged down. She has given me the energy and motivation to keep going.

Table of Contents

List of Figures

List of Tables

Abstract

Fuzzy logic is used for set point determination in a linear quadratic tracking problem for the U.S. Air Force's C-18 Cruise Missile Mission Control Aircraft (CMMCA). The CMMCA's mission is to radar track cruise missiles (CM) during test flights. Because of the complexity of the CM flight profiles, maintaining radar coverage at all times is very difficult. A simulation was constructed to develop fuzzy logic and optimal controls to provide 100% radar coverage. The fuzzy logic was used in a series of intervals to determine the set point. The set point calculation's fuzzy logic balanced CMMCA maneuvering based on present and future CM positions. Three different future times were used: 60, 90 and 120 seconds ahead, and the performance for each time was compared. The final form of the fuzzy logic provided varying radar coverage at each look ahead time for a complex CM flight path (CM in 20 degrees of bank) 1850 seconds long. At 120 seconds look ahead time, the coverage was 100%. When the same profile was performed with the CM in 30 degrees of bank, coverage was degraded, and 90 seconds look ahead performed best.

A FUZZY LOGIC OPTIMAL SOLUTION TO THE CMMCA TRACKING PROBLEM

I. Introduction

1.1 Background

The U.S. Air Force currently uses a large number of aircraft during each cruise missile (CM) test flight. To reduce the number of required aircraft, the Air Force introduced the C-18 Cruise Missile Mission Control Aircraft (CMMCA). The CMMCA is modified with an AN/APG-63 air-to-air radar similar to that in the McDonnell-Douglas F-15. The CMMCA is designed to completely replace all other aircraft currently flown in support of each CM test flight. The CMMCA's mission is to track the CM for its entire flight, keeping the CM within the radar field of view as much as possible, providing continuous telemetry reception and deconflicting the airspace from any intruders.

Prior to entering the test range, the CM and CMMCA fly generally straight paths at medium altitudes. During this portion of the flight, the CMMCA has no difficulty radar tracking the missile. Upon entering the test range, the CM descends to below 1000 feet and begins a long, elaborate series of maneuvers consisting of turns in all directions. The turns are usually linked by short straight segments; some turns, however, have no straight segments between them (see Figure 1).

The missile flies at a nearly constant speed of 400 knots (nautical miles (nm) per hour), while the CMMCA can vary its speed between 320 and 480 knots. However, because speed changes result in greater fuel consumption, the CMMCA should fly as close to a constant airspeed as possible. The CMMCA has an operational limit of 30 degrees of bank, as well as a minimum/maximum "g" capability of 3/4 to 2 g's. Exceeding any of these limits can put excessive stresses on the

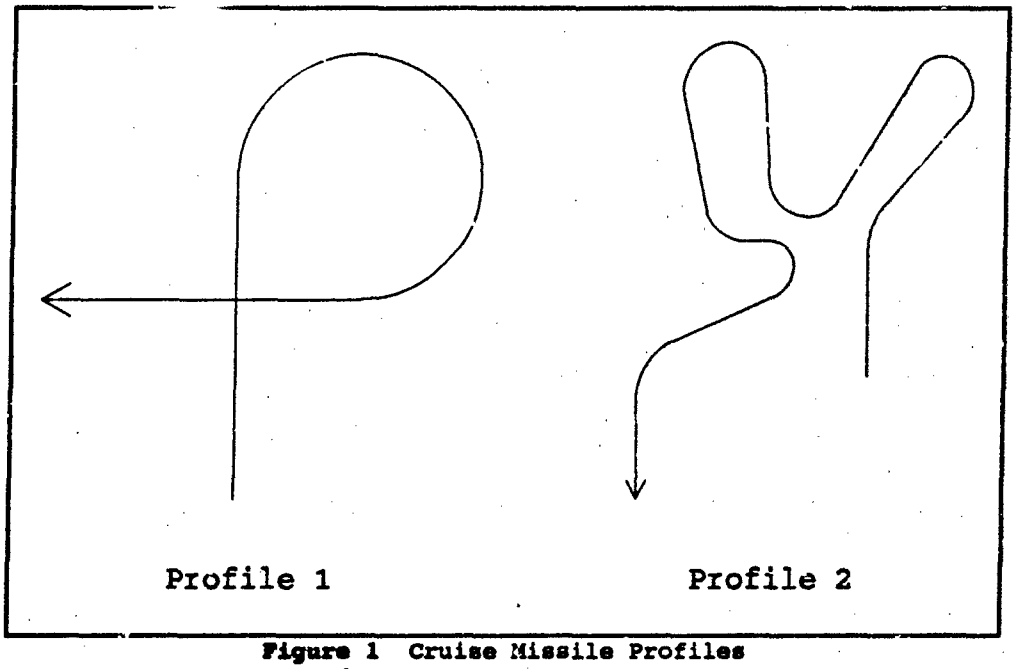

Figure 1 Cruise Missile Profiles

airframe. Although the CM is capable of large bank angles, during test flights it generally turns at smaller bank angles of 15-20 degrees.

The CMMCA radar field of view is a cone centered on the fuselage and 60 degrees to either side of center. The minimum range is 5 nm and maximum range is 15 nm (see Figure 2). The CMMCA attempts to fly a nominal distance r_0 in trail of the CM (directly astern) with the missile at the 12 o'clock position. Although the value of r_0 is not critical, the initial value used will be a slant range of 10 nm (the center of the radar envelope). Similarly, the nominal *look angle* (angle from the nose of the CMMCA to the CM) will be zero degrees in azimuth.

1.1.1 Possible Maneuvers of the CMMCA. Several factors conspire to complicate this problem. They include the restricted radar envelope of the CMMCA and the large spacing of the two vehicles relative to their turn radii.

As soon as the missile begins an extended turn, the CMMCA is out of position and must maneuver to return to position. The CMMCA can continue to follow the same ground track as the missile, can follow the missile until losing radar contact and then turn blindly toward the

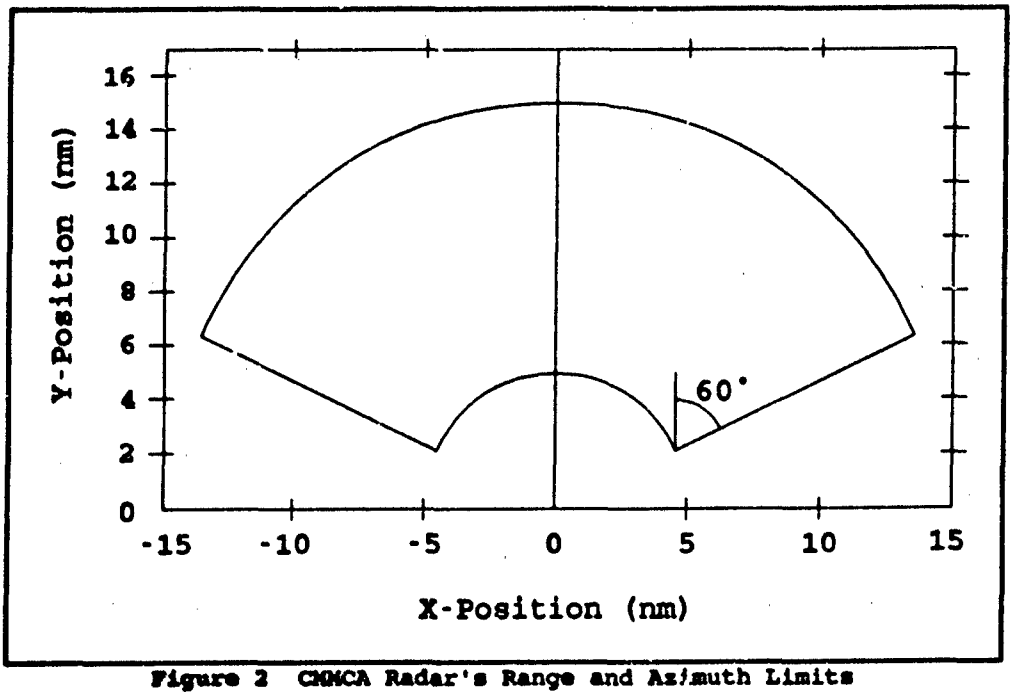

Figure 2 CMMCA Radar's Range and Azimuth Limits

missile, or follow by pointing directly at the missile.

Suppose the missile makes a 180 degree turn and the CMMCA flies as close as possible to the CM flight path. Radar contact will be lost after the CMMCA has flown 6.5 nm and reacquired at 13.5 nm, at a range of 11.2 nm (see Figure 3a). If instead, the CMMCA continues straight ahead until nearly losing radar contact, then flies its best turn through 180 degrees, radar contact will be lost after 6.5 nm and reacquired at 11.0 nm, barely outside minimum range (see Figure 3b).

Now suppose the CMMCA simply points at the CM while the missile makes a 180 degree turn. The distance between the two will decrease and the angle between their headings will increase until, after the CMMCA has flown 5.5 nm, it loses radar contact at minimum range. If the CMMCA then flies its tightest turn, it will still not reacquire the missile for another six nm. Instead of being at r_0, the CMMCA will be at minimum range, only five miles in trail (see Figure 4).

The CMMCA can also change its airspeed in an attempt to maintain radar coverage. Doing so will allow the CMMCA to follow the missile ground track because of its reduced turn radius, but the missile may

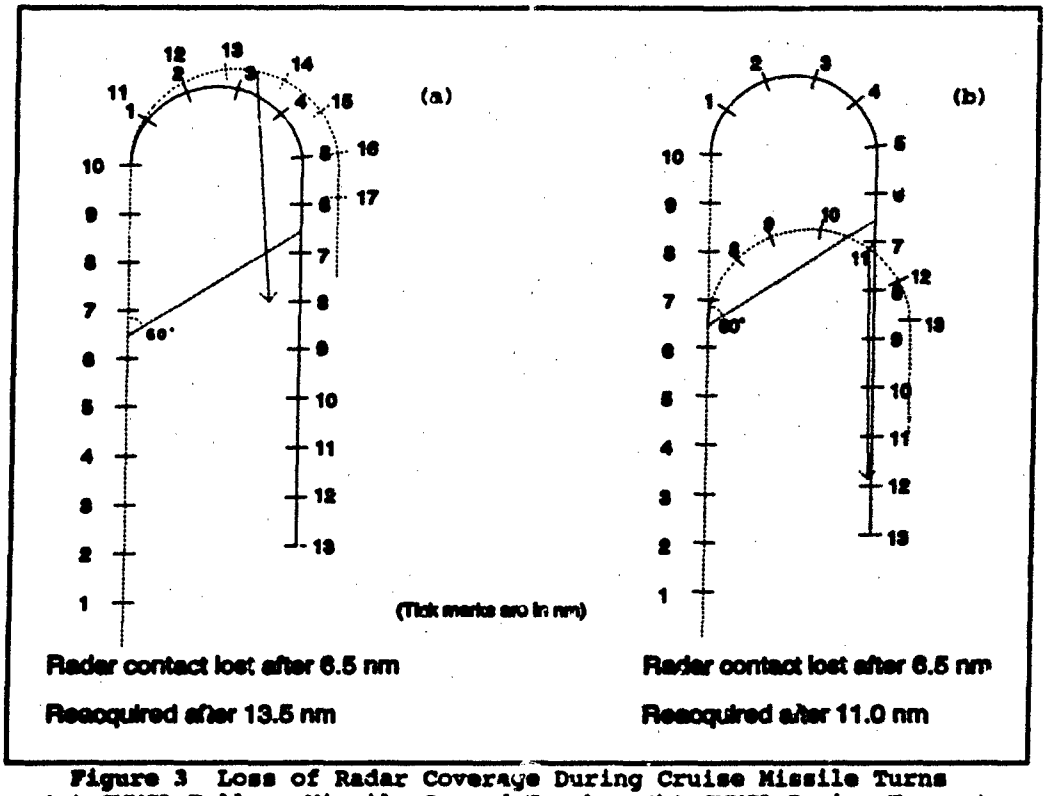

Figure 3 Loss of Radar Coverage During Cruise Missile Turns (a) CMMCP Follows Missile Ground Track. (b) CMMCA Begins Turn at Radar Azimuth Limit

still fly out of the radar coverage. The major reason for minimizing airspeed changes is because of the effects on CMMCA fuel consumption. Another important reason is that it is simpler, faster and more efficient to increase or decrease CMMCA to CM spacing by turning rather than by changing airspeed. However, if the CM speeds up or slows down for an extended period of time, then the CMMCA will probably need to change speed also.

Another complication results if the CMMCA attempts to *stand-off* and let the CM maneuver well in front of it, because the CMMCA must greatly increase its range or turn well away to avoid overflying the missile. Because the CMMCA will cover over 16 nm while the CM executes a single turn of 360 degrees, and the span of minimum to maximum radar range is only 5 to 15 nm, the CMMCA will almost certainly lose radar coverage during the missile's maneuvering. Moreover, reaching a stand-off position and resuming track after the stand-off is complete are

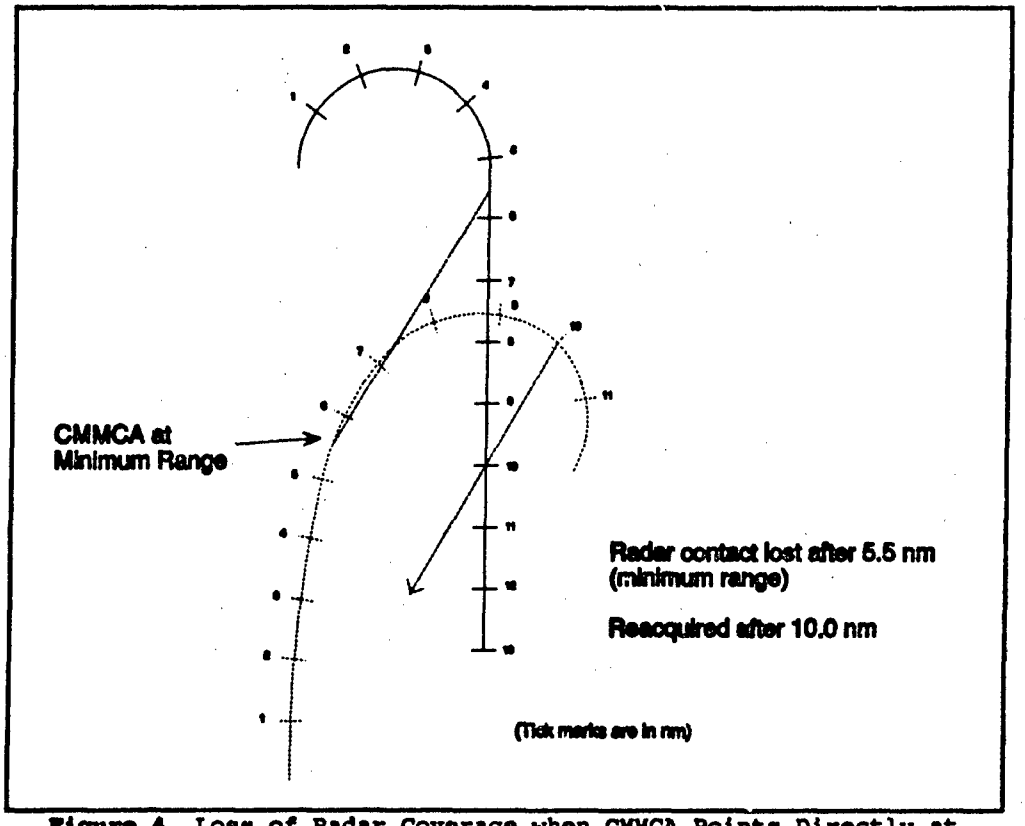

Figure 4 Loss of Radar Coverage when CMMCA Points Directly at Missile as Long as Possible

formidable problems that are more easily avoided by less complicated CMMCA flight paths.

Due to its recent acquisition, the pilots and crew of the CMMCA have not yet worked out robust procedures and techniques for solving the tracking problem, nor is help available in the form of autopilot or other control methods. In fact, pilots still rely on trial and error, experience, and luck to follow the CM through its convoluted and elaborate flight profile.

Three attempts have been made so far to solve the CMMCA tracking problem. Heavner, in his MS thesis, applied dynamic programming to the CMMCA, but was only able to work out optimal solutions for a single small turn. The complexity was too great for larger flight paths, and computer memory was rapidly exhausted (8:36-38).

Garton approached the CMMCA problem from a different direction; he wrote a Fortran based simulation using gradient search techniques to minimize a penalty function, and an initial straight and level flight path. The penalty function incorporated variable weights and penalized deviations of the CM from the center of the CMMCA's radar coverage. Although performing well for simple maneuvering, the most complex maneuver considered by Garton was a single 270 degree turn with the CM in only 20 degrees of bank. (6:48-9).

Hachman's thesis was an extension and refinement of Garton's work. He modified the penalty function by adjusting the weighting constants, and tried several different initial CMMCA flight paths. He then used a gradient search procedure to produce optimal weightings for the penalty function. Hachman achieved good results in most cases, although he still didn't achieve completely feasible CMMCA flight profiles (7:62).

1.2 Purpose of the Research

CMMCA pilots require the means to continuously radar track the CM. Previous research on this subject has provided only partial answers. These answers apply either to very small portions of the flight profile, or extend to larger portions but with gaps in the radar coverage. Also, these earlier results only provided pilots with optimal or near optimal CMMCA ground tracks for specific missile profiles. They did not provide control laws that could be used to track a general profile. The intent of this thesis will be to formulate basic control laws that will provide a means for the CMMCA to track the CM through all phases of its flight, maintain 100% radar coverage, and not exceed the operating limits of the CMMCA.

1.3 Problem Statement

Cruise Missile (CM) test flights require the CMMCA to maintain constant radar contact on the CM while not exceeding aircraft limits. However, no method exists to provide guidance and control information to the CMMCA. This thesis will derive optimal control laws for the CMMCA

that provide maximum radar coverage and remain within operational limits.

1.4 Assumptions and Scope

Several assumptions and limitations are required in this thesis to narrow the scope and complexity. They are:

1. The equations of state and motion are based on a flat, non-rotating earth.

2. Control laws are determined using a continuous-discrete model. CMMCA and CM movements are calculated in discrete time intervals of one second. Within each time interval, the calculations are carried out in continuous time based on the system state at the beginning of the interval. Chapter 2 has a detailed explanation of this model.

3. The CM is at a constant speed of 400 knots for the entire flight.

4. The CM is in level flight, although its altitude actually varies somewhat as it follows land contours.

5. The CMMCA is level at 29,000 feet.

6. No attempt was made to use exact performance figures for the CMMCA or CM. Instead, the analysis uses nominal values for CMMCA and CM performance and limitations, taken from the previous works.

7. The CMMCA is restricted to bank angles less than 30 degrees.

8. The CMMCA's true airspeed (TAS) is held constant at 400 knots.

9. CMMCA and CM are unaffected by winds or other atmospheric effects.

10. The CMMCA is always in coordinated flight (no sideslip).

11. CM flight path data is assumed to consist of map coordinates only. No CM heading or start/stop turn point information is used. All CM maneuvers have been simplified to straight lines and circular arcs.

12. All CMMCA and missile parameters and states are considered exact rather than stochastic (i.e., no random errors in observations and no process noise).

13. The CMMCA radar field of view is a cone centered on the fuselage and 60 degrees to either side of center. The minimum range is 5 nm and maximum range is 15 nm.

14. Radar performance is uniform throughout the entire region of coverage.

15. Both vehicles are modeled as a point mass.

16. Future knowledge of the CM flight path is perfect.

All depictions of ground tracks throughout this thesis will be two dimensional for ease of interpretation. However, the three dimensional slant range and radar look angles were actually used in all calculations.

II. Literature Review

2.1 Introduction

Aircraft and missile guidance and control has received a great deal of attention in the previous half century. The subject covers a broad range of topics from air-to-air combat to missile and airplane intercepts. The field of guidance and control also extends to rendezvous on other aircraft and to tracking problems, where one aircraft follows another at some specified position.

The purpose of guidance and control is to generate some type of rendezvous or intercept with a second, possibly maneuvering, aircraft (the target). The problem is how to *best* apply control inputs to the controlled aircraft (the pursuer). In many cases, the desired outcome is to bring pursuer and target close enough for the pursuer to employ its weapons/ordnance. In the case of rendezvous and tracking, however, target destruction is not normally a goal. For a rendezvous, the pursuer attempts to arrive at the same position as the target and at the same velocity. For tracking, when defined for controlling one aircraft with respect to another, the pursuer attempts to maintain some predetermined position relative to the target.

The CMMCA control problem is fundamentally a tracking problem. Therefore, for this thesis the scope will be limited to those problems where the objective is to maintain a position relative to the target. The term *tracking* will be synonymous throughout with *CMMCA tracking*, i.e. with the CMMCA maintaining a preset position a nominal distance in trail of the CM. This definition is a very narrow definition of the normal meaning of tracking, and is adopted here to prevent having to invent a new term for the CMMCA tracking problem.

Although there is a vast amount of material available discussing missiles intercepting aircraft, the literature is much sparser for rendezvous problems, and nearly silent when it comes to tracking. Any

investigation of the CMMCA tracking problem must begin with a review of the literature concerning intercept and tracking problems. Specifically, this proposal examines dynamic programming, penalty function methods, differential game theory, and optimal control theory.

2.2 Dynamic Programming

An approach that is sometimes useful for this class of problem is known as dynamic programming (DP). According to Winston, DP "obtains solutions by working backward from the end of a problem toward the beginning, thus breaking up a large unwieldy problem into a series of smaller, more tractable problems" (19:715). Dynamic programming is an easily implemented approach, however, it has severe known limitations: a linear increase in problem size causes an exponential increase in complexity. It is highly computer intensive, even with small problems. Heavner ran into exactly this problem in his DP implementation of the CMMCA problem (8:36). Winston states the five basic principles of DP:

1. The problem can be divided into stages with a decision required at each stage.
2. Each stage has a number of states associated with it.
3. Each decision transforms the current state into the next.
4. The optimal decision for each stage must not depend on previously reached states or previously chosen decisions.
5. There is some *cost* function to be minimized/maximized that depends only on the states. (19:722)

The term stage in most cases refers to the discrete time intervals into which the problem is broken. The state is the full set of information needed at each stage to make the correct decision. The state is often the position and velocity of each aircraft, however, the choice of states is arbitrary since there is an infinite number of states that lead to the same result (9:53). In practice, the states are chosen to make the problem simpler.

DP is especially applicable to the discrete time quadratic regulator problem. For this problem, the linear system is given by the state equation:

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k), \qquad x(t_0) = x_0$$

where A and B are matrices, $x(t_k)$ is the state vector at time t_k, $u(t_k)$ is the control at time t_k, and x_0 is the initial condition. The problem is to find $u(t_k)$ that minimizes the cost J, where

$$J = x(t_N) P_f x(t_N) + \sum_{i=0}^{N-1} [x^T(t_i) Q x(t_i)] + r \sum_{i=0}^{N-1} u^2(t_i)$$

Here a superscript "T" represents a transpose operation, and P_f, Q and r are relative weightings of the cost terms. P_f penalizes deviation at the final time $t = t_N$, Q penalizes deviation from the desired state during the time interval $[t_0, t_N]$ and r is the control cost (9:243-4).

Heavner, in his MS thesis, applied DP to the CMMCA by using time intervals of 6 seconds and a four-state implementation consisting of the CMMCA's position and velocity. He used CMMCA bank angle and thrust as control variables (the inputs to the system). He then related the state of the CMMCA at the next time period to its state at the current time period plus the change caused by the controls (8:18).

Heavner minimized a cost functional based on CM deviations from the center of the CMMCA's radar coverage. Instead of a quadratic cost, his cost had the form

$$J = \int_{t_0}^{t_f} L[X(t), U(t)]dt + O[X(t_f), t_f] \qquad (1)$$

where L penalized the sum of range and azimuth deviations during the flight, and O penalized missile deviations during the end game (8:19,72). Heavner discretized J by converting the integral to a summation and explored several different time steps.

Heavner's results were limited to CM maneuvers of a single turn of 90 degrees or less. Computer memory requirements increased so rapidly that the solution soon became impossible, a problem noted by Winston (19:754). Heavner further reported that the results were highly dependent on the time interval used, and suggests that his penalty function could be significantly improved (8:37-8).

2.3 Penalty Function Method (Optimization)

Garton applied a penalty function to the CMMCA tracking problem. That is, he used the form of equation 1, but with a significantly more sophisticated penalty function. Instead of simply summing the range and azimuth deviations, as Heavner did, Garton used

$$J[V,\alpha] = \sum_{i=0}^{N} Wt_i \, J_i$$

$$J_i = [r(t_i) - r_0]^2 + W_1[\theta(t_i) - \theta_0]^2 + W_3\left(\frac{V_i - U_0}{\mu}\right)^{2K_3} + W_4\left(\frac{\alpha_i}{\beta_0}\right)^{2K_4}$$

where N is the number of time intervals, r(t) and $\theta(t)$ are the range and azimuth from the CMMCA to the CM (r_0 and θ_0 are their nominal values), V and α are the velocity and bank angle of the CMMCA (U_0 is the nominal velocity), W_1 through W_4, μ, K_3, and K_4 are weights to be determined, and the Wt_i are quadrature integration weights (6:19-23).

Garton calculated J based on an initial (guessed) CMMCA flight path. His starting flight path was for the CMMCA to fly straight and level. He then used a gradient search procedure to reduce J. To compute the gradient of J, he first formed a maneuver vector consisting of the CMMCA speed and bank angle at each time interval (a 2N × 1 vector). He then calculated the gradient from the partial derivatives of J with respect to speed and bank. He sampled three J values along this gradient and fit a parabola to the points, then found the minimum of this parabola. This became his next maneuver vector, and the process repeated until J converged to a minimum (6:24-33).

Garton successfully applied this method for the CM straight and level, and for the CM in 20 degrees of bank for a 90 degree turn. When applied to a CM turn of 270 degrees, however, the algorithm exceeded the CMMCA's bank angle restriction (6:43-49).

Hachman started with Garton's algorithm and scaled the penalty function components so that any deviation of the same magnitude produced the same penalty (7:28-30). He used two initial CMMCA flight paths:

1) straight and level, the same as Garton, and 2) the CMMCA flying exactly the same flight path as the CM at the desired distance behind the missile. Hachman also set all parameters to Garton's computed values, and set the weighting factors to one. Hachman used four CM profiles:

1. a 180 degree turn.
2. a 270 degree turn.
3. a 270 degree turn to the right, followed immediately by a 270 degree turn to the left.
4. an actual test flight fragment with sever straight segments and six turns of between 45 and 200 degrees (see Figure 5) (7:30-35).

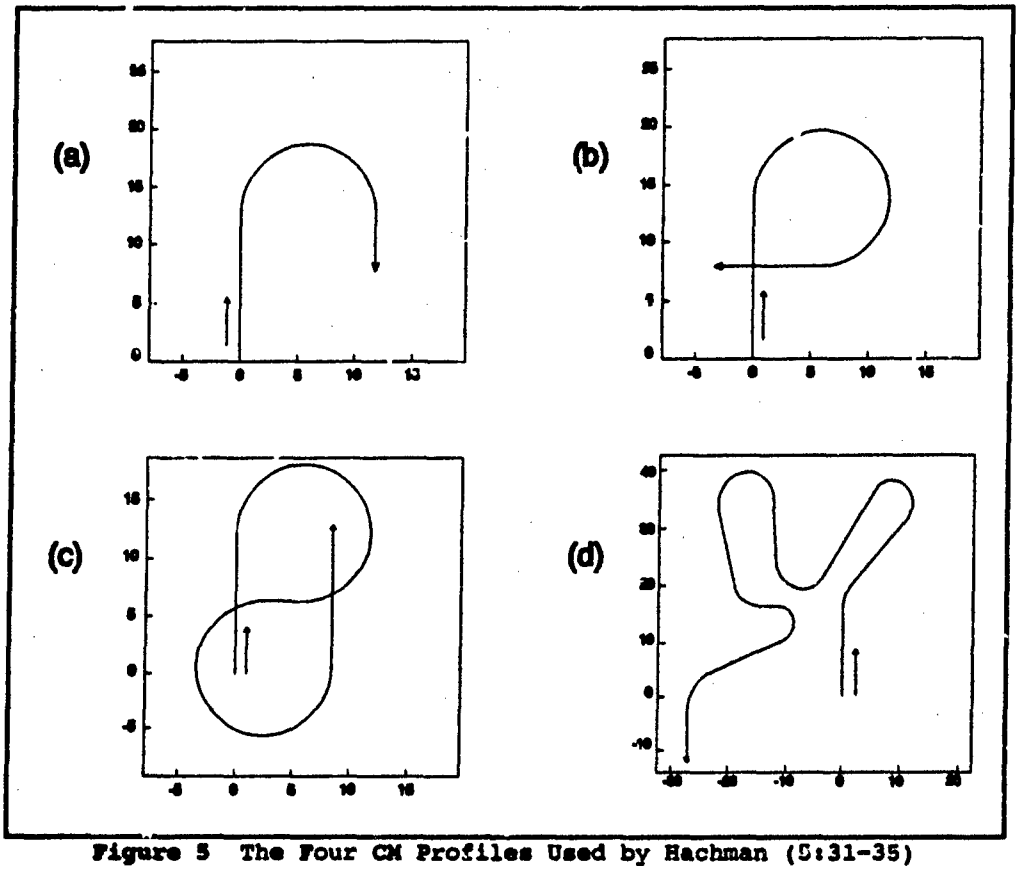

Figure 5 The Four CM Profiles Used by Hachman (5:31-35)

Hachman's results are shown in Table 1. Profiles one and two met all CMMCA constraints and achieved 100% radar coverage. Profiles three and four both showed significant degradation in radar coverage when the initial CMMCA ground track was straight and level. When the CMMCA's initial path was in trail of the CM, the results were nearly optimal.

Table 1 Hachman's Preliminary Results for Four CM Profiles (5:50)

Flight Path	Initial Path	% Radar Coverage	% of time in CMMCA Limits
1	Straight	100	100
	Trailing	100	100
2	Straight	100	100
	Trailing	100	100
3	Straight	74.15	94.56
	Trailing	100	100
4	Straight	37.41	100
	Trailing	64.68	97.90

Hachman then implemented a 2^4 full factorial design on profile three. He used the three weighting factors and the CMMCA to CM nominal range r_0 as his design parameters in an attempt to maximize the percent radar coverage. His intent was to produce a set of optimal weights and an optimal nominal range.

The factorial design proceeded by selecting 2^4 design points, each with a different combination of weights and r_0. A linear regression was performed on the results, producing a first order *response surface*. The gradient of this surface was taken at its best (but still suboptimal point). An examination of the responses along the gradient from this point was used to find a local maximum. This local maximum was then used as the center for another set of design points.

This search procedure was iterated through two gradient searches. Hachman discovered on the second gradient search that he was finding no improvement in radar coverage (the response), and therefore he stopped. He selected as optimal that set of weights that had given him the best response. (7:38-40,49-55).

This point of the gradient search on profile three was to provide optimal weights that could then be applied to profile four. This did, if fact, allow Hachman to achieve good results for CM profile four. The

radar coverage improved from 63% to 100% and CMMCA limits were maintained at 98%.

Hachman tried two other methods to improve profile four's results. He first tried double precision variables, which resulted in an improvement of about 1% in CMMCA limits with no loss of radar coverage. An extreme drawback of this method was that each run required nearly seven hours of CPU time on a VAX model 6420 computer.

Hachman reported his best success came from breaking the profile into two pieces, roughly equal in length. In this case, no change occurred from the CMMCA in a trailing initial position, but the straight and level initial position improved greatly (see Table 2) (7:56-9).

Table 2 Hachman's Results for CM Profile Four Using Three Different Approaches (5:56,57,59)

Method	Initial Path	% Radar Coverage	% of time in CMMCA Limits
Optimal Weights From Profile 3	Straight	53.85	100
	Trailing	100	97.9
Double Precision Variables	Straight	46.15	100
	Trailing	100	98.25
Splitting Flight Path into Two Segments	Straight	97.55	100
	Trailing	100	97.67

Hachman found several major difficulties with his method. The amount of CPU time increased rapidly as CM profiles grew longer. Profile four took approximately three hours for one run (of 150 iterations). When split into two pieces, each piece still took 50 minutes of CPU time (7:59).

Another major problem Hachman found was difficulty in achieving convergence of the gradient search technique to minimize the penalty function (7:65). This was part of the cause of the significant increases in computer time required to find the optimal trajectory.

In summary, Hachman's method requires a starting CMMCA flight path, which is currently produced by guesswork. The number of iterations to produce a minimum penalty function is large, and time required appears to increase exponentially with CM flight path length. Worst of all, the output from this method is simply a ground track for the CMMCA to follow. There is no way to implement this form of the solution by autopilot or to use the actual aircraft data such as range and bearing to the missile.

2.4 Differential Game Theory

Two player, zero-sum games are those in which two competitors are engaged in a contest that defines a winner and a loser. Games in which the movements of the players can be described by differential equations are known as differential games. Air combat involves two adversaries maneuvering with the offensive adversary (pursuer) attempting to shoot down the defender (target). The arena of air-to-air engagements between pursuer and target is an example of a two player, zero-sum differential game.

Bryson and Ho discuss linear-quadratic pursuit-evasion games (1:282-289). The mathematical basis of all differential games is the Maximum Principle. Pursuer's and evader's equations of motion are:

$$\dot{x}_p = F_p x_p + G_p u, \qquad x_p(t_0) \text{ given}$$

$$\dot{x}_e = F_e x_e + G_e v, \qquad x_e(t_0) \text{ given}$$

where the subscripts p and e represent pursuer and evader, u and v are the pursuer's and evader's control inputs, and F and G are matrices relating the change of state to the current state and controls.

Menon considered the problem of two aircraft, one strictly offensive and the other strictly defensive (13:27-28). The goal of the pursuer is to drive the distance between them to zero, while the defender seeks to maximize the distance. With U and W the pursuer's and defender's *pseudo-control* inputs, respectively, Z the relative position between pursuer and defender, S a positive semidefinite matrix penalizing separation at the final time t_f, and A and B positive

definite matrices representing acceleration capabilities of the pursuer and defender, respectively, then the cost functional is

$$J = \min_{U} \max_{W} \left[\frac{1}{2} Z^T(t_f) S_f Z(t_f) + \int_{t_0}^{t_f} [U^T A U - W^T B W] dt\right]$$

(13:29). The pursuer attempts to minimize the cost of maneuvering while driving the range to zero; conversely, the target attempts to minimize the cost of maneuvering while trying to increase the range as much as possible.

U and W are pseudo-controls because they are obtained by transforming the actual aircraft controls, (i.e., throttles, "g's," and bank) into their components in inertial 3-space. This also has the effect of transforming the nonlinear vehicle dynamics into a linear, time-invariant form. Menon's model is formulated in terms of the inertial position state variables and their various derivatives; the solution obtained is in a feedback form. After forming the Hamiltonian, Menon uses the variational calculus to derive a closed form solution of the control laws. The inverse transformation results in a non-linear feedback solution in the original coordinates (13:27-29).

Menon reports results for two scenarios. Both involve fighter type aircraft of nearly equal capabilities. In the first, the evader is successfully intercepted in 14.5 seconds. In the second scenario, the pursuer fails to intercept and the evader escapes (13:30-2).

Inclusion of a weapon's range makes an air-combat model even more realistic, and can be accomplished by assuming a spherical weapons envelope of radius R_w. As soon as the distance between the players decreases to R_w, the problem is solved and the target is deemed destroyed.

Menon & Duke devised an even more realistic weapons envelope that took the form of an arbitrary 3-dimensional shape. They started with Menon's model described above and then expanded it by inclusion of an arbitrary weapons envelope. They cite simple examples such as a cone of limited size directly forward of the pursuer, and a prolate spheroid

(football shape) centered at the pursuer's center of mass (14:449-450). The cone-shaped weapons envelope is very similar to the CMMCA's radar coverage.

An additional factor in intercept problems is the specification of the final time -- the time at which the intercept occurs. Final time is a critical value; variations can significantly change the outcome of the intercept or rendezvous. Menon and Duke were able to leave final time unspecified, then derive it from the equations of motion and from energy conservation laws (14:453).

As in Menon's earlier paper, variational calculus is applied to the Hamiltonian equation to derive a closed form solution to the control laws. The same transformation and inverse transformation are used to go from the nonlinear dynamics to a linear time-invariant form, then back to the original coordinates (14:451-3).

Menon and Duke report the result from one engagement scenario. As in Menon's paper, both aircraft are fighter types of nearly equal capabilities except the pursuer's top speed is almost twice the evader's. The evader begins 2 km in front of and 5 km to the right of the pursuer, with the evader's heading 90 degrees away from the pursuer. The evader begins a turn away and tries to out run the pursuer. The pursuer begins an immediate turn and completes the intercept after 51 seconds. The effect of the weapon's envelope can be seen in the altitude acceleration history of the pursuer, where it causes several large oscillations (14:454-5).

Differential game theory is a promising avenue of approach; its assumptions are reasonable and the use of a weapons envelope is remarkably similar to that of the radar environment in which the CMMCA operates. The major drawback to using game theory for the CMMCA tracking problem is exactly its most basic feature. The evader is maneuvering in reaction to the pursuer, attempting to maximize relative range, yet in the CMMCA problem, the CM follows a preset course.

2.5 *Optimal Control Theory*

The theory of optimal control has been developed to derive a set of input controls to drive a dynamic system in some *optimal* fashion, that is, in a way that minimizes a cost. In the linear quadratic gaussian formulation, the cost is based on a performance index that is defined as deviations from some desired state, and the amount of control input provided to the system. The resulting optimal control law is easily implemented on nearly any computer and the results are rapidly available. Further, for a linear system, most of the computations can be done *off-line*, that is, prior to actually applying them to the system (10:52).

In general, a system (plant) can be described by the non-linear dynamical equation

$$\dot{x}(t) = f(x,u,t)$$

where x(t) is an n dimensional state vector and u is an m dimensional control input vector. Associated with the system is a performance or cost index

$$J = \Phi(x(t_f),t_f) + \int_{t_0}^{t_f} L(x(t),u(t),t)dt$$

over the time interval $[t_0,t_f]$. The cost $\Phi(x(t_f),t_f)$ depends on the final state and final time, while the weighting function L(x,u,t) depends on the state and input at intermediate times. Optimal control is the problem of finding "the input $u^*(t)$ on the time interval $[t_0,t_f]$ that drives the plant along a trajectory $x^*(t)$ such that the cost function" is minimized (10:151).

In many cases, the plant can be adequately expressed with a linear time-varying system of equations

$$\dot{x} = A(t)x(t) + B(t)u(t) \qquad (2)$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$.

Associated with the system is a quadratic performance index

$$J = \frac{1}{2} x^T(t_f) S(t_f) x(t_f) + \frac{1}{2} \int_{t_o}^{t_f} [x^T Q(t) x + u^T R(t) u] dt \qquad (3)$$

over the time interval $[t_o, t_f]$. In the most general formulation, "weighting matrices $S(t_f)$ and $Q(t)$ are symmetric and positive semidefinite, and $R(t)$ is symmetric and positive definite" (10:180).

This system is an example of a linear-quadratic (LQ) regulator. "A *regulator* is designed to keep a stationary system within an acceptable deviation from a reference condition using acceptable amounts of control" (1:148). Another type of system is the *terminal controller*, which is "designed to bring a system close to desired conditions at terminal time (which may or may not be specified) while exhibiting acceptable behavior on the way" (1:148).

Using state and costate equations and boundary conditions, the optimal control is expressible in terms of A, B, and R^{-1}. However, u^* does not depend on the state $x(t)$, making this an *open loop* controller. If some perturbation moves the system off the predicted course, the controller will not compensate. Hence a *closed loop* controller that can react to system perturbations is preferred.

To develop a closed loop, feedback control structure, the method of Lagrange multipliers is often used. Start by adjoining the state equation (2) to the control cost (3) using a (vector) Lagrange multiplier $\lambda(t) \in \mathbb{R}^n$ (9:226-7),

$$J = \frac{1}{2} x^T S x + \int_{t_o}^{t_f} [\tfrac{1}{2} x^T Q x + \tfrac{1}{2} u^T R u + \lambda^T (Ax + Bu - \dot{x})] dt$$

$$= \frac{1}{2} x^T S x + \int_{t_o}^{t_f} [H - \lambda^T \dot{x}] dt$$

where the Hamiltonian H is defined as

$$H = \frac{1}{2}[x^T Q x + u^T R u] + \lambda^T(Ax + Bu)$$

Using results from the calculus of variations, the state and costate equations are (10:180-181,189-191)

$$\dot{x} = \frac{\partial H}{\partial \lambda} = Ax + Bu$$

$$-\dot{\lambda} = \frac{\partial H}{\partial x} = Qx + A^T u$$

and the stationarity condition is

$$0 = \frac{\partial H}{\partial u} = Ru + B^T \lambda \qquad (4)$$

Solving equation (4) for u(t),

$$u(t) = -R^{-1}B^T\lambda(t))$$

Further, we must have

$$\lambda(t_f) = S(t_f)x(t_f)$$

Now assume this same relation between $x(t_f)$ and $\lambda(t_f)$ holds for all $t \in [t_o, t_f]$:

$$\lambda(t) = -S(t)\, x(t)$$

If there exists such a matrix S(t), then this assumption will have been valid. Differentiate the costate equation and substitute the state and costate equations to get (dropping the explicit time dependence notation)

$$-\dot{S}x = (A^T S + SA - SBR^{-1}B^T S + Q)x$$

and since this must be true for all t

$$-\dot{S} = A^T S + SA - SBR^{-1}B^T S + Q, \qquad t \leq t_f$$

This is the matrix Riccati equation. If the Riccati equation has a solution, then the assumption was valid and therefore the optimal control is

$$u^*(t) = -R^{-1}B^TSx^*(t)$$

According to Kailath, the solution to the Riccati equation exists and is unique if the system is both controllable and observable (9:231). Defining the Kalman gain as

$$K(t) = R^{-1}B^TS(t)$$

the optimal control is

$$u^*(t) = -K(t)\cdot x^*(t)$$

The Riccati equation is solved backwards in time for S(t). Note that this can be done *off-line*, since the state x(t) does not appear in the equation. Then the Kalman gain can be computed and stored, to be applied later (10:190).

Lewis discusses several other results:

1. Even if A, B, Q, and R are time-invariant, K(t) still varies with time.

2. The plant dynamics are

$$\dot{x} = (A - BK)x$$

For any initial condition $x(t_0)$, this will yield the optimal state trajectory $x^*(t)$.

3. If the optimal control is used on $[t_o,t_f]$, then the performance index is

$$J(t) = \tfrac{1}{2}x^T(t)\cdot S(t)\cdot x(t)$$

The advantage of this is that cost can be computed in advance. If the cost is too high, the weighting matrices Q, R, or $S(t_f)$ can be changed (10:191).

Computers are, of course, used to solve the above equations, and the digital implementation lends itself to sampled data rather than continuous solutions. In the discrete version, the optimal control is a piecewise constant function, that is, changed only at the beginning of each discrete time step. The continuous equations for a linear time

invariant plant (such as the CMMCA) are easily made discrete (12:42,43;10:63-65). The interval from t_0 to t_f is divided into N periods of length T (the sampling period). The plant equation becomes

$$x(t_{k+1}) = \Phi(t_{k+1},t_k)\cdot x(t_k) + \Gamma(t_{k+1},t_k)\cdot u(t_k)$$

where Φ and Γ, the sampled plant and control matrices, are given by

$$\Phi(t_{k+1},t_k) = e^{AT}$$

$$\Gamma(t_{k+1},t_k) = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1},\tau)\,B(\tau)\,d\tau$$

This is the *continuous-discrete* control law formulation. The optimal control u^*_k is discrete: constant during each time interval and switched to a new value only between steps. u^*_k is computed from the above equations for the optimal gain, but using the sampled matrices in place of the system matrices. The state $x^*(t)$ is continuous and is found by summing over all complete time intervals, and integrating during the current one:

$$x(t) = e^{A(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{A(t-\tau)}B\,d\tau\,u_k, \qquad t_k \le t < t_{k+1}$$

2.6 Fuzzy Logic and Fuzzy Control Systems

2.6.1 Introduction. Generally, the output response to a set of inputs is derived from one or more mathematical equations, from a set of rules (expert system or Boolean logic), or from a transfer function. Many times, however, the resulting algorithms are so complex as to be completely or essentially useless. Applications requiring real time solutions may not have a solution calculable in real time (e.g. convex optimization problems). To arrive at mathematically tractable, real time solutions, it is common to make simplifying assumptions. Doing so, however, often means that the resulting errors are excessive, rendering the solution unusable.

Fuzzy logic is a way around these difficulties, and is often vastly simpler and much more intuitive. Consider the fuzzy rule: "If

brake temperature is Warm AND speed is Not Very Fast, then brake pressure is Slightly Decreased" in a car's anti-lock brake system vs the conventional formulation: "if (200 is less than brake temperature is less than 280) AND (speed is less than 45 RPM), then brake pressure is 190" (5:58). In this case, the degrees of warm and speed will determine the amount of brake pressure. In essence, this one fuzzy logic rule supplants many rules in a conventional rule-based system.

According to Chiu et al., fuzzy logic control has "excellent robustness characteristics, perhaps because the inherent imprecision, or generality ... is well suited to imprecise systems whose behavior is known only in the large" (3:43). Cox likens a fuzzy model to a parallel processor: "All the rules that have any truth in their premises will fire and contribute to the output fuzzy set," i.e. to the control variables (5:60).

The important and distinguishing feature of a fuzzy rule system is its reliance on human experience and intuition. A fuzzy control algorithm "can be regarded as a set of heuristic decision rules or 'rules of thumb'" (11:65). As guidelines for when to use fuzzy logic, Cox suggests: "when one or more control variables are continuous; when a mathematical model of the process does not exist, or exists but is too complex to be evaluated [rapidly or takes up too much chip memory] ..., when high ambient noise levels must be dealt with, ...; and perhaps above all, when an expert is available who can specify the rules underlying the system behavior and the fuzzy sets that represent the characteristics of each variable" (5:58).

2.6.2 Fuzzy Sets. The concept of a fuzzy set is fundamental to fuzzy logic. A fuzzy set is any set in which elements may have *in-between* membership in that set. Unlike *normal* sets where an element is either in or not in a set, an element may belong only partially to a fuzzy set. The degree of set membership is a real number from 0 to 1 with 0 denoting no membership and 1 representing full set membership.

With the notion of fuzzy sets in hand, the next step is to consider each variable. Li and Lau's *universe of discourse* is the set of all possible values a variable can take (i.e. its domain). Having specified the domain, it is generally a simple matter to divide the domain into subsets based on experience and the current situation. For example, temperature may be readily classified as Cool, Tepid, Warm, etc. and the actual temperatures corresponding to Cool will vary depending on whether the system under consideration is a fusion reactor core or a liquid helium containment vessel.

Knowing the universe of discourse, a mapping from analog variable values to set membership values is needed. The exact shape of the membership function is unimportant; any arbitrary choice is acceptable, although simplicity usually drives the choice. Figure 6 shows two commonly used shapes: trapezoidal and triangular. Although a trapezoidal shape is slightly more complicated, it captures the fact that a variable, over a restricted range, is often fully in one set and not in any others.

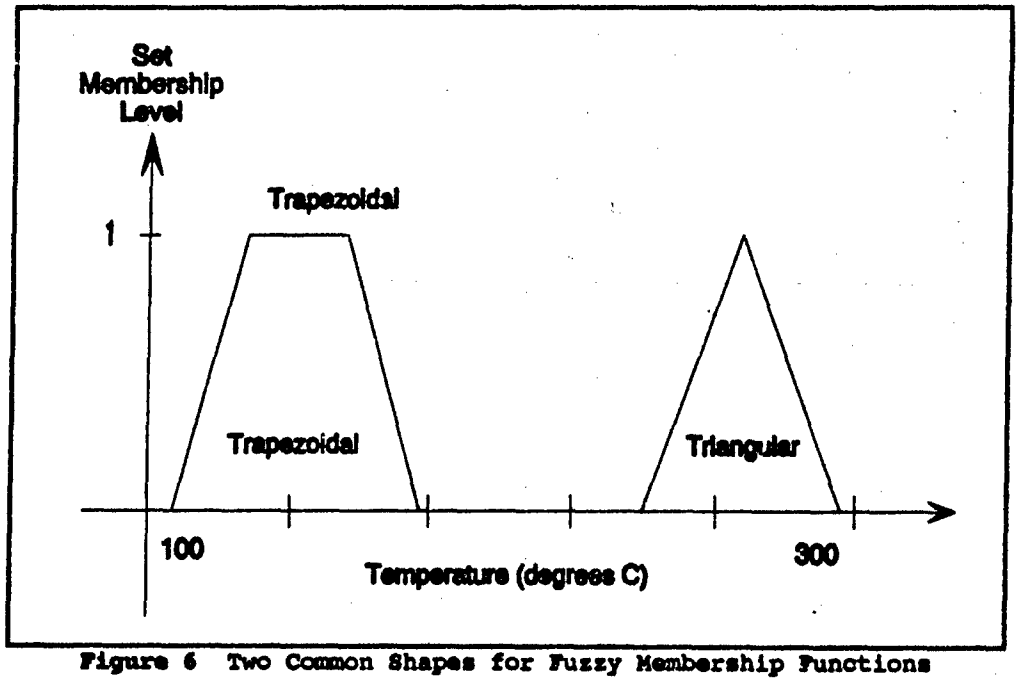

Figure 6 Two Common Shapes for Fuzzy Membership Functions

Let $u_A(x)$ represent the degree that element "x" belongs to set A (the set membership function). Then continuing the brake example from above, a temperature of 100 may belong to the sets Cool, Tepid, and Warm with

$$u_{Cool}(100) = .3,$$
$$u_{Tepid}(100) = .6,$$
$$u_{Warm}(100) = .1.$$

This step of assigning set membership is *fuzzification*. Note the overlap of the three sets. This is a key feature of fuzzy logic and shows where the *parallel processing* paradigm arises, since a variable is often a member of multiple sets. Also, using descriptive names for sets instead of nested conditional statements (a series of "if ... then" constructs) allows a more intuitive understanding of the decision logic being derived.

The mapping from analog variable value to set membership could be accomplished directly. However, to make the mapping to set membership level generic, each variable is first mapped into the same interval, e.g., [-5,5], and *linguistic sets* are then defined. Linguistic sets are merely standard name replacements for variable specific set names such as Cool or Short. A generalized *fuzzification* of any variable can be made with the linguistic sets Large Positive (LP), Small Positive (SP), Zero (ZR), Small Negative (SN), and Large Negative (LN) (see Figure 7). Hot, for example, might correspond to the linguistic set LP. This classification can be expanded further with additional sets such as Medium Positive or Very Negative, or reduced by eliminating, for example, all Negative sets (11:65). This scaled value is then used in a look-up table for each linguistic set (see Table 3).

Cox provides some rules of thumb for defining linguistic sets:

1. The number of sets should generally be an odd number between five and nine.
2. Each set should overlap with its neighbors, since this overlap is what gives a fuzzy controller its smooth and stable surface.

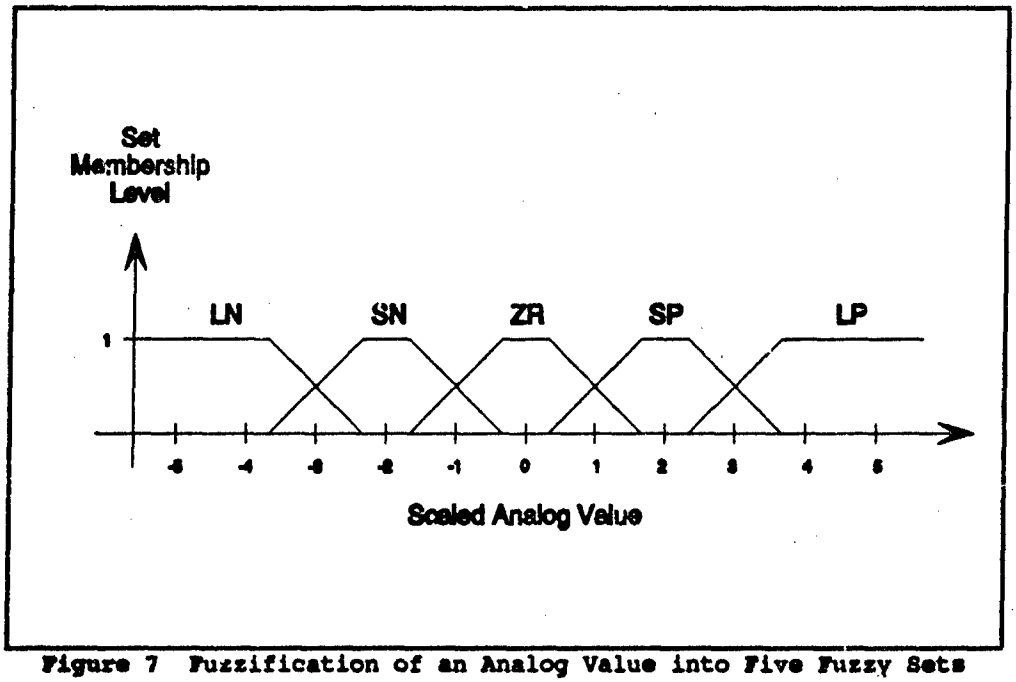

Figure 7 Fuzzification of an Analog Value into Five Fuzzy Sets

Table 3 Membership Matrix Table

Linguistic Sets	Quantized Levels										
	-5	-4	-3	-2	-1	0	1	2	3	4	5
LP	0	0	0	0	0	0	0	.0	.5	1	1
SP	0	0	0	0	0	0	.5	1	.5	0	0
ZR	0	0	0	0	.5	1	.5	0	0	0	0
SN	0	0	.5	1	.5	0	0	0	0	0	0
LN	1	1	.5	0	0	0	0	0	0	0	0

3. The overlap should be 10 to 50 percent of the neighboring space, but the sum of the membership values at any point must not exceed one.
4. The density of the linguistic sets should be highest near the optimal point. (5:61).

2.6.3 Fuzzy Logic. Once linguistic sets are decided on for all input and output variables, a full set of logic rules is needed to relate the system inputs to the outputs. The rules are devised using the linguistic sets and the three operators AND, OR, and NOT. While a

large number of rules may be required, the rules will be far simpler and fewer in number than in the corresponding non-fuzzy formulation.

Fuzzy set logic is based on the following definitions of the basic logic operations:

1. The union of set A and set B (logical OR) is

   $u(A \text{ OR } B) = \max[u_A(x), u_B(x)]$

2. The intersection of set A and set B (logical AND) is

   $u(A \text{ AND } B) = \min[u_A(x), u_B(x)]$

3. The complement of set A (logical NOT) is

   $u(\text{NOT } A) = 1 - u_A(x)$

2.6.4 Fuzzy Control. With the linguistic sets defined for the input and output variables of a given system, and with the necessary logic rules formulated to define specific input-output relationships, fuzzy logic next derives the overall system output by combining the output of each rule. As an example, suppose the following three logic rules are in effect:

1. If Input 1 is ZR AND Input 2 is LN, then Output is SP,
2. If Input 1 is SP AND Input 2 is SN, then Output is ZR,
3. If Input 1 is LN AND Input 2 is SN, then Output is SN.

Suppose also that the current scaled value of Input 1 is 0.8 and the value of input 2 is -2.5. This is shown in Figure 8.

Rule 1 produces 0.7 from input 1 and 0.2 from input 2. Application of the rule's logical AND then produces min[0.7,0.2] = 0.2, which is used to truncate the output SP trapezoid. Rule 2 produces 0.3 which truncates the ZR output at the 0.3 level. Rule 3 produces 0.0, leading to no contributed output.

Once the outputs for each rule are determined, they must be combined into a single output value. The combination of individual outputs is commonly made from one of two basic methods: the composite maximum or the composite moment (or centroid). The choice of which method to use is somewhat arbitrary, but is often based on simplicity of implementation.

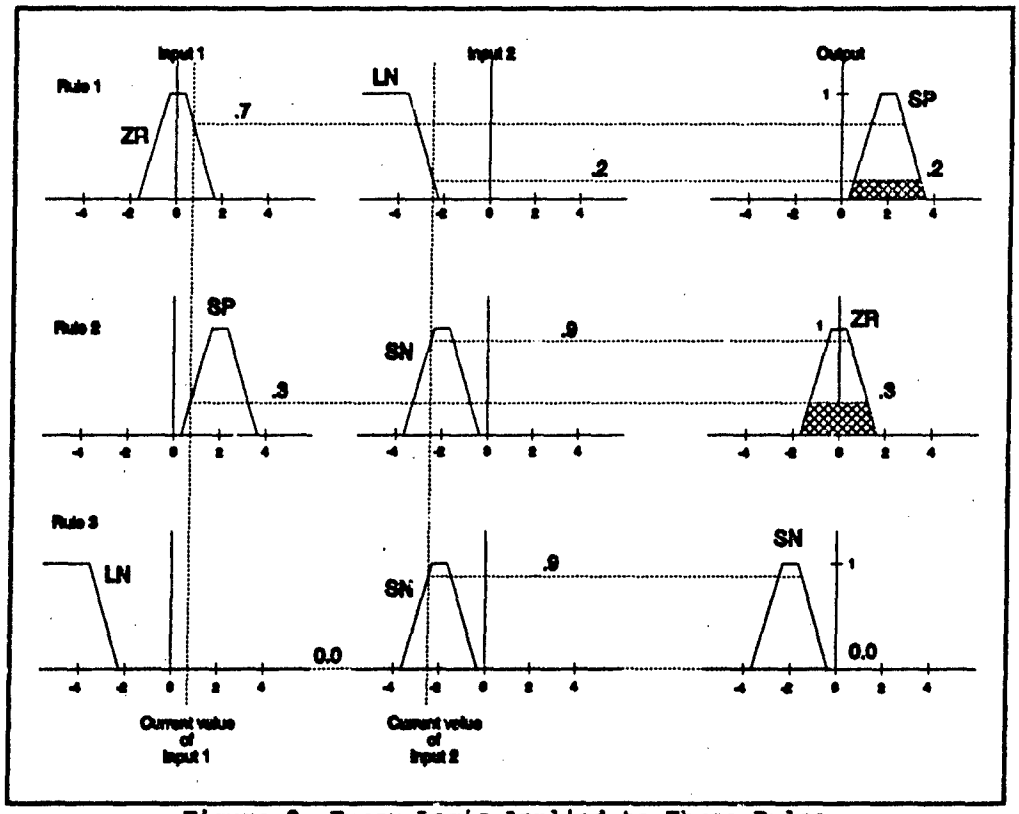

Figure 8 Fuzzy Logic Applied to Three Rules

The composite maximum is the simplest method, and consists of simply choosing that output having the maximum value. Because this ignores the results of all other *firing* logic rules, a significant amount of information can be lost and the value of the fuzzy approach is reduced. As Figure 9 shows, the centroid method provides an output value of 0.0, which is just the midpoint of the maximum output (ZR from Rule 2) and ignores the output of Rule 1 (SP), which is nearly as large as the chosen output.

The composite moment method is somewhat more complicated, but its advantage lies in the fact that each firing rule contributes to the output. The various outputs at each point are combined by a logical OR operation. This has the effect of taking the maximum value of all outputs. The centroid of the combined output is then the final value of the output. In essence, each output is weighted by the value of its respective rule. As shown in Figure 9, the centroid method provides an

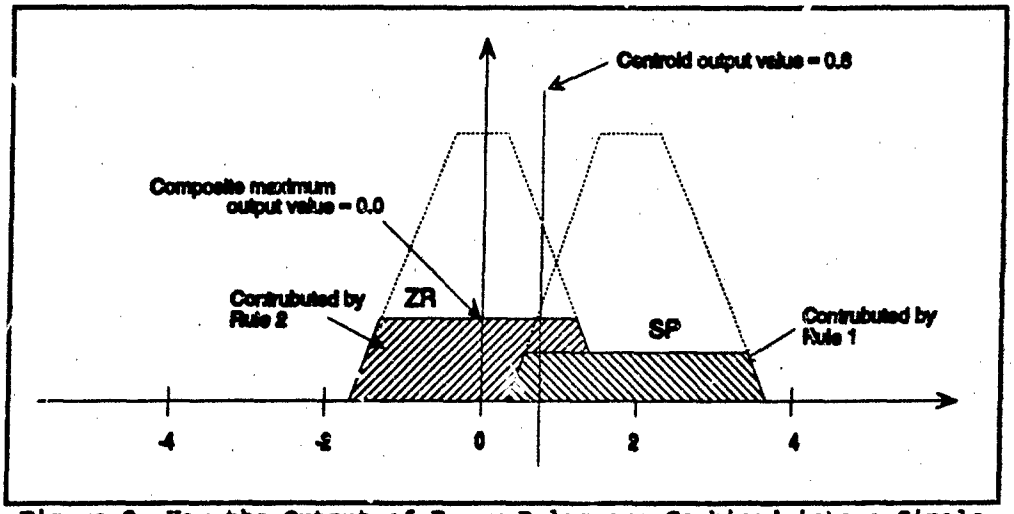

Figure 9 How the Output of Fuzzy Rules are Combined into a Single Output Value

output value of 0.8.

The centroid method is often preferred because it uses all the available information, producing a "result that is sensitive to all the rules, ... hence the results tend to move smoothly across the control surface" (5:61). The result of either the composite maximum or the composite moment method is a (scaled) value of the output. This must then be converted to an absolute (analog) value by the appropriate transformation.

While the fuzzy logic procedures can be implemented directly and computed each time they are needed, Li and Lau (11:67) recommend using a look-up table for speed and simplicity. The table is entered with the scaled input values and provides the corresponding scaled value of the output variable (see Table 4).

As an example, suppose brake temperature and speed are the inputs and brake pressure is the output. Given a temperature of 255 °C and a speed of 247 RPM, these variables scale, for example, to 1.3 and 3.7, respectively. Entering Table 4 with these values for inputs 1 and 2 results in an output value of -3.7 (scaled), which is then transformed to its final value (e.g. 7.9 psi).

Table 4 Sample Look-Up Table (Li:68)

Input 1	Input 2										
	-5	-4	-3	-2	-1	0	1	2	3	4	5
-5	5	5	4	4	3	3	3	2	1	1	0
-4	5	5	4	4	3	3	2	1	1	-1	-1
-3	5	5	4	3	2	2	1	0	0	-2	-2
-2	5	4	3	3	2	1	1	0	-1	-3	-3
-1	4	4	3	2	1	1	0	-1	-2	-3	-3
0	4	3	3	2	1	0	-1	-2	-3	-3	-4
1	3	3	2	1	0	-1	-1	-2	-3	-4	-4
2	3	3	1	0	0	-1	-2	-3	-3	-4	-5
3	1	1	-1	-1	-1	-2	-3	-4	-4	-5	-5
4	1	1	-1	-1	-1	-2	-3	-4	-4	-5	-5
5	1	0	-1	-2	-2	-2	-3	-4	-5	-5	-5

Fuzzy controllers have a problem as the current state of the system approaches the desired state and the control approaches zero. The fuzzy controller may then not provide optimal control, but instead result in overshoots and oscillations about the desired position due to overcontrol. The solution to this problem is to have two different fuzzy logic decision tables, one for coarse control and another for fine control (11:68).

2.6.5 Results of Fuzzy Controllers. Li and Lau compare fuzzy control algorithms to two other types of algorithms for a servomotor system controlling the velocity and angular position of its shaft. The conventional controllers are a proportional-integral (PI) controller and a Model Reference Adaptive Control (MRAC) controller. The fuzzy system used coarse and fine controllers; it may sometimes be necessary to add a third decision table to dynamically change the gain to prevent instability problems (11:70).

The fuzzy controller outperformed the PI controller and is as good as the MRAC controller. It has one-half the settling time of the MRAC and two-fifths that of the PI controller. Li and Lau caution not to

expect an optimal response except for certain ranges of the inputs due to the tuning and scaling that was done to the controller. Expanding the operating region means adjusting the parameters.

Importantly, fuzzy theory is not yet mature enough to provide firm guidance in how to shape or by how much to overlap the various membership functions. Too much overlap will result in too many rules simultaneously being applied, while too little overlap will make the derivation of the look-up table difficult (11:71).

Chiu et al. report on the use of fuzzy control logic for controlling the roll rate and load alleviation control of the Advanced Technology Wing (ATW). The system enforced torsion moment bounds without any significant degradation in roll rate performance. They then varied the plant parameters by 50% to reduce stability, resulting in small overshoots in roll rate and slightly increased torsion moment oscillations. "Nevertheless, the control system still maintained good, stable roll response and stringent enforcement of the torsion moment bounds" (3:47).

They further observe that the simple set of rules derived from qualitative considerations provided a highly robust control system. Some of the bounds are readily derived from physical constraints, while others are initially estimated and then adjusted in an iterative process to arrive at the desired response (3:47).

Their final statement is an excellent reminder of the potential dangers of fuzzy logic: "the greatest drawback of fuzzy control is the lack of rigorous stability and robustness analysis techniques. ... The nonlinear nature of fuzzy control affords enhanced performance at the expense of analytic tractability" (3:47).

2.7 Conclusion

There are several very valid methods that can be applied to the CMMCA tracking problem. Dynamic programming can, in theory, solve any control problem, but the implementation is flawed by the tremendous amount of computer memory required for any but the simplest problem.

The penalty function method of Garton and Hachman achieved good results, but used vast amounts of computer CPU time and did not provide any means for implementing them on aircraft autopilots.

Differential game theory, especially with the introduction of a weapons envelope similar to the CMMCA's radar cone, seems to provide an alternative. Unlike differential game theory, where the evader is actively attempting to defeat the pursuer, the CM is not evading. Hence the extra complexity is not needed, but removing it reduces the problem essentially to an optimal control problem. Optimal control offers easy implementation, quick results, and the ability to provide actual control information to the CMMCA.

Fuzzy logic, used as an adjunct, would help to implement whatever solution method is chosen while taking into account the experience and knowledge of the implementer. It provides implementation of that experience in a way that remains intuitive and relatively uncomplicated, but is still sophisticated enough to capture the needed level of detail.

III. Methodology

3.1 Introduction

This chapter describes the procedures used in this thesis to solve the CMMCA tracking problem. The solution came from a linear quadratic (LQ) optimal control approach implemented in a nearly real time computer simulation. Located within the LQ framework are all the performance data for both the CMMCA and the CM, system parameters and the actual mechanism for computing the optimal CMMCA flight path. A fuzzy controller evaluates current and future geometry and provides the desired set point. A simulation was used for debugging, model verification, and performance analysis.

The first section discusses background material needed to formulate the LQ solution and set up the simulation. This includes the solution algorithm and associated implementation details. Then, because the proper computer tools were critically important to the development of a working simulation, both the computer hardware and software used in this thesis will be covered. Next is a detailed accounting of the inputs to the computer software, as well as a description of the origin and value of the specific numerical values used. Then the different scenarios used in this thesis will be detailed. This will be followed by a description of the form and content of the simulation output. Finally, the details of the simulation itself will be described.

3.2 Background and Setup

Since this thesis is concerned exclusively with the solution of the CMMCA tracking problem, it would certainly be possible to model the flight capabilities of the CMMCA and the CM precisely. This would involve extensive research into various performance and design specifications and parameters of each vehicle. Such an approach has been eschewed in this thesis, both because it needlessly complicates the basic issue, and because such exactness is neither required nor desired.

A more general and less exacting framework was used. In order to concentrate on the theory and problem solution, it was deemed more productive to minimize time spent on gathering exact values for maximum roll rates, "g" capabilities, thrust and drag, etc. In place of exact data, approximate values were selected so that both CMMCA and CM simulated performance closely approximated that of the real vehicles.

3.2.1 States of the System. One of the basic issues for any LQ approach is defining the states of the system. Since both the CM and the CMMCA are free to move in three dimensions, each has six degrees of freedom. For example, the CMMCA has a position and angle of rotation along or about each of three coordinate axes. However, for this problem, the number of states can be reduced to fewer than the number of degrees of freedom, due to the constraints imposed on the two vehicles, and the amount of information needed to carry out the simulation.

The CM can be described using only its position in two dimensions as a function of time. Then it is a simple matter to calculate velocities, accelerations and bank angles. This was further simplified by several assumptions. The first was that the CM is either in straight and level flight, or turning at a constant bank angle, and that the CM makes instantaneous transitions. Second, the CM is flying at a constant airspeed of 400 kts (675.11 feet/sec). Third, the CM maintains a constant altitude of 1000 feet above sea level. Finally, the CM follows the preprogrammed profile at all times.

The set of states for the CMMCA cannot be as simple. However, because of the dynamics of coordinated aircraft flight, the six degrees of freedom are not independent. In particular, the aircraft pitch (angle of nose above/below the horizon) and yaw (heading) can be readily computed from the three velocity components. Hence the seven scalar values of position, velocity and bank angle can completely describe the CMMCA.

3.2.2 Coordinate Axes. To make these seven states convenient for use in an LQ formulation, it is necessary to add the rate of change of

the bank angle (the bank angle rate). This 8-state vector for the CMMCA then consists of the positions and velocities in an X, Y, and Z coordinate system, the bank angle and the bank angle rate. The coordinate system is a simple inertial Cartesian system (oriented North, East,and down) fixed to the Earth with its origin at the initial position of the CMMCA (see Figure 10).

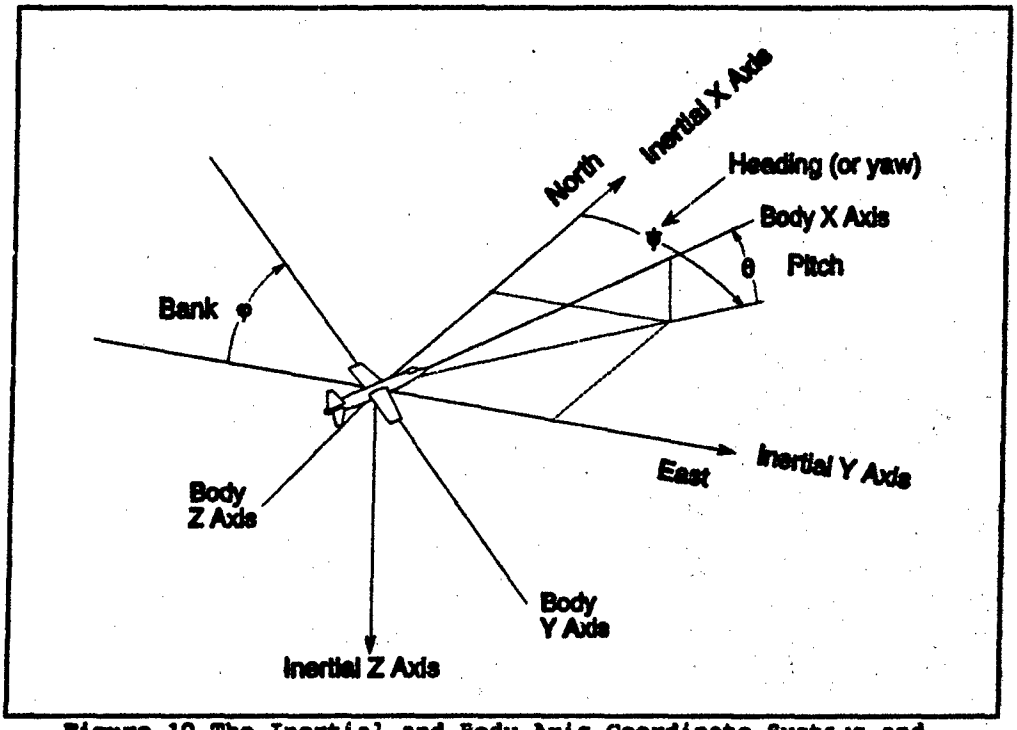

Figure 10 The Inertial and Body Axis Coordinate Systems and Euler Angles (ψ, θ and ϕ)

To simplify matters, the CM state vector is made identical to that of the CMMCA. In the case of the CM, however, the values of bank angle and bank angle rate inside the state vector are not needed or calculated, but merely carried as place holders.

Two other coordinate systems are necessary for certain calculations. They are a body centered inertial system and a body axis system. Although these systems can be defined for any body, they are only needed here for the CMMCA.

The body centered inertial system is exactly the same as the fixed inertial XYZ coordinates, except that the origin is located at the

instantaneous position of the center of mass of the CMMCA. That is, it is an inertial XYZ system with axes parallel to the fixed system and with origin translated to the CMMCA's present position.

The body axis system is also shown in Figure 10. The origin is located at the center of mass of the CMMCA. The positive X-axis is defined along the direction of the fuselage. The positive Y-axis is defined perpendicular to the X-axis and outward along the right wing. The positive Z-axis is then defined as perpendicular to both the X and Y axes, and outward through the bottom of the aircraft.

Acceleration along the X body axis is the change in the speed of the CMMCA (if no sideslip is present). Acceleration along the Y body axis is sideslip. Sideslip is generally avoided, especially in low performance aircraft such as the CMMCA; it is assumed zero in this thesis. Z body axis acceleration is caused by the lift of the wings and is the "g" force. The upper "g" limit is bounded by the structural limits of the CMMCA, while the lower "g" limit is usually a function of oil and fuel starvation to the engines and passenger discomfort.

3.2.3 The Control Variables. With the states and coordinate axes defined, the next item of interest is to determine the control variables. These variables are the inputs to the systems; by varying them the system is driven to the desired state. In problems involving aircraft, the natural control inputs are the movements of the throttles, control stick or yoke, and rudders.

These controls together fully specify the motion of any conventional fixed wing aircraft, including the CMMCA. (The rudder pedals can be ignored here, since their effect is usually to eliminate sideslip and hence to make all turns coordinated.) The throttle controls the acceleration of the aircraft along the body X-axis. The yoke controls two different accelerations. Fore and aft movement changes the lift produced by the wings and thus controls acceleration along the body Z-axis. Rotation of the yoke controls the roll rate or rotation about the body X-axis.

Although the above three controls are the most *natural*, they are not the ones used in this thesis. Instead, four accelerations are used: three along the fixed inertial coordinates axes and one being the rate of change of the bank angle. Even though the X, Y and Z inertial accelerations are not directly attributable to physical events (such as moving the throttles), they can be transformed quite simply into accelerations corresponding to throttle and yoke movements.

Inertial accelerations are used mainly to simply the bookkeeping and mathematical tractability of the LQ formulation, and to maintain the intuitiveness of the simulation from the viewpoint of an external observer. Also, accelerations are used rather than actual physical movements of aircraft control structures. This is also done for simplification. It is relatively straightforward, but quite tedious, to convert accelerations to yoke and throttle movements.

3.2.4 The Linear Quadratic Approach. The eight states and the four controls for the CMMCA can be described respectively by

$$X = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix}; \qquad U = \begin{bmatrix} u_x \\ u_y \\ u_z \\ u_\phi \end{bmatrix}$$

The dynamics of both the CMMCA and the CM (the plant) can then be expressed in one of two control law formulations: discrete or continuous. This thesis uses the discrete state equation

$$X(k+1) = \Phi_s \cdot X(k) + \Gamma_s \cdot U(k)$$

Here X(k) is the state in the kth time interval, U(k) is the control applied in the kth interval, and Φ_s and Γ_s are the time invariant state transfer function and control input matrices, respectively.

Because the CMMCA flight is, in reality, a continuous process, the plant was first formulated in the continuous form of the state equation

$$\dot{X}(t) = A(t) \cdot X(t) + B(t) \cdot U(t)$$

The sampled matrices Φ_s and Γ_s are extracted from A(t) and B(t).

The control inputs U(t) would also actually be applied in continuous time, however, this thesis used a discrete control input U(k). It should be noted that the discrete time interval used (a sample time of one second) was small compared to aircraft response time and thus closely approximated a continuous control input.

The continuous time state equation is:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\omega \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \omega \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \omega \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \\ u_\phi \end{bmatrix}$$

The parameter ω represents the roll response of the aircraft. A large value of ω means any bank angle change command rapidly affects the bank angle, while a small value of ω means there will be some delay before a command to roll right or left is realized.

The cost of control, J, was given in the previous chapter as

$$J = \frac{1}{2}\, x^T(t_f) S(t_f) x(t_f) + \frac{1}{2} \int_{t_0}^{t_f} [x^T Q(t) x + u^T R(t) u]\, dt$$

over the time interval $[t_0, t_f]$. Weighting matrices $S(t_f)$ and Q are symmetric, positive semidefinite; R(t) is symmetric, positive definite.

Initial values of Q, R and S were needed to proceed with the LQ formulation. The initial values chosen for Q and R were

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this thesis, a steady state approach to the LQ problem was followed. This meant an infinite final time was used to calculate the

optimal controls and the Riccati equation solution. With an infinite final time, the differential Riccati equation became the algebraic Riccati equation, and the Riccati solution S and the Kalman gain K were constant matrices. The steady state solution tends to provide asymptotic convergence to the desired state.

3.2.5 Time Intervals. Unlike a classical regulator problem, where the desired state of the system is constant over time, the desired CMMCA state is continuously changing as the CM maneuvers. In fact, the desired CMMCA state – the set point – is a complex function of the current CMMCA state, the current and future CM states, and the range and angles from the CMMCA to the CM.

To successfully deal with the myriad factors involved, three different time intervals were defined (the simulation itself uses additional time intervals internally). These were the sample time, the final time update time, and the look ahead time. Figure 11 illustrates these time intervals. Sample time was discussed in the previous section and was set to the constant value of one second.

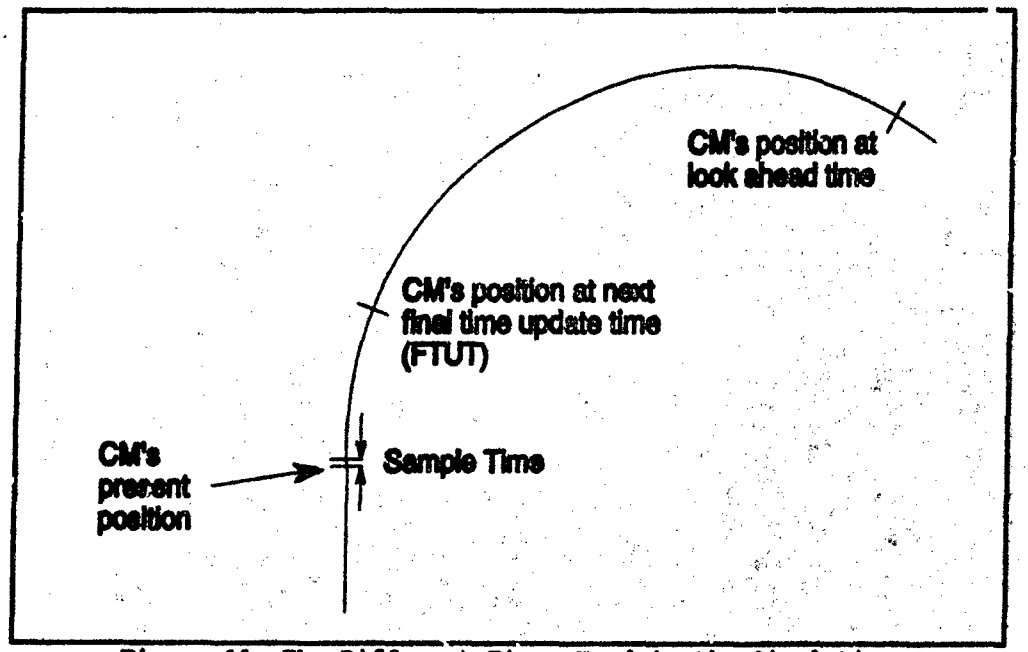

Figure 11 The Different Times Used in the Simulation

The CMMCA flight path was broken into a large number of small segments, the end of which were called the final time update time (FTUT). In essence, a moving final time was being used. The length of each FTUT segment was chosen to be ten seconds, although it would be possible to dynamically calculate the FTUT segment length based on CM present and future maneuvering.

At the start of the FTUT interval, a set point for the CMMCA was computed, and the optimal controls were applied to drive the CMMCA towards that point. During each interval, the set point was constant. At FTUT, the next set point was calculated, and the CMMCA was then driven to that set point. This *leap-frogging* of set points was repeated for the entire flight profile.

Figure 12 shows this in detail. The large "X" is the next set point for that FTUT interval. The CMMCA is driven towards the set point for the duration of the present FTUT interval. However, because of roll lags and finite accelerations, the CMMCA will only make it to the location denoted by a "O." From this point, the next set point is computed, and the tracking begins again. Thus the CMMCA is driven repeatedly through a large number of successive tracking solutions.

To exploit the knowledge of the missile's flight path, it was necessary to use the future CM state. The time interval into the future was called the look ahead time (t_ahead). How far into the future to look was certainly not clear a priori. Therefore, three different look ahead times were examined: 60, 90 and 120 seconds.

3.2.6 Fuzzy Logic and Set Points. As mentioned in the previous section, the entire simulation consisted of a series of time intervals during which the CMMCA was driven to the next set point. The determination of the next CMMCA set point is far from trivial. Fuzzy logic was used to calculate where the set point should be. It took into account numerous, although by no means all possible factors.

The location of the next set point was based on what was the best maneuver for the CMMCA, i.e., the best course of action given the states

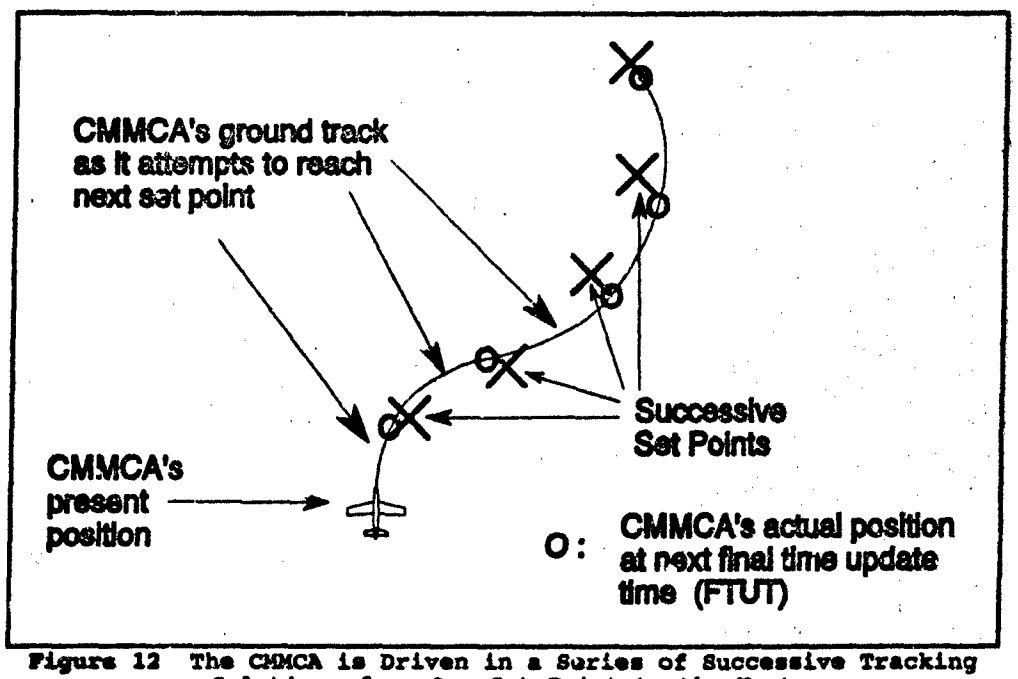

Figure 12 The CMMCA is Driven in a Series of Successive Tracking Solutions from One Set Point to the Next

of the CMMCA and CM at the present time and in the future at the look ahead time. To do this, the fuzzy logic combined all the factors considered into a bank angle between the maximum allowable left bank and the maximum right bank. While the same logic with the same factors could have been used to compute a speed change, no speed changes were considered in this thesis.

Since speed and altitude were held constant during the simulation, the only control left was the CMMCA bank angle. Hence the fuzzy logic was used to choose a bank angle for the CMMCA. The set point would then be located along this theoretical flight path at a distance corresponding to a flight time of FTUT. Figure 13 shows the locus of set points that the CMMCA could reach (the dashed line). The large "X's" represent the CMMCA position if the maximum bank angle of 30 degrees is used, or if the CMMCA does not turn. As an example, the set point with the CMMCA in 12 degrees of right bank is also shown.

The bank computed by the fuzzy logic is not necessarily related to the bank angle generated by the optimal control. The logic's bank angle is just a convenient way to determine the set point in terms of where

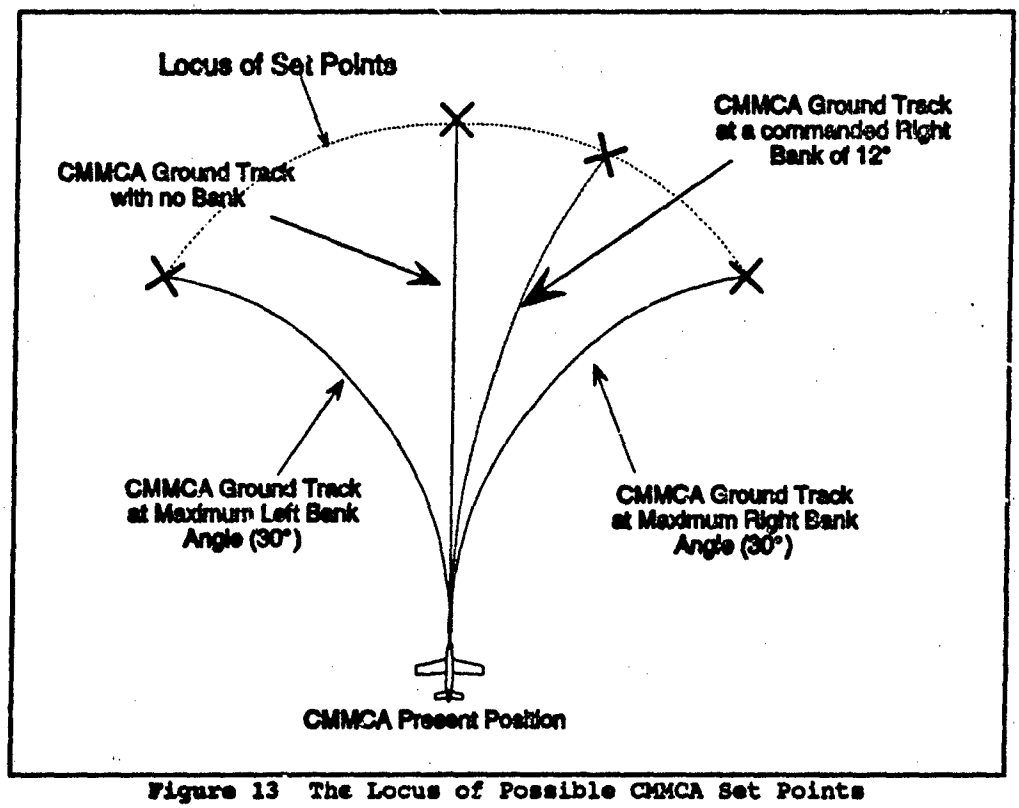

Figure 13 The Locus of Possible CMMCA Set Points

the CMMCA is capable of reaching. It was then a simple matter to transform this bank angle into a point in inertial space.

These values were idealized in the sense that current bank angle was not considered. In other words, it was assumed the CMMCA would instantaneously achieve the desired bank angle. The lag caused by the roll rate led to minor errors between actual and set point states. However, this was counteracted by two factors. First, the next set point was recomputed at each FTUT independent of past calculations, thus freeing current errors from future considerations. Second, the lag occurred on both ends. Although the CMMCA took a finite amount of time rolling into bank, for example, at the start of a FTUT interval, the same lag occurred at the next FTUT interval, effectively causing the CMMCA to correct for the lag time.

The fuzzy logic for computing the required bank angle is applied in two parts. The first part determines required bank angle using the

current values of the CMMCA and CM states. This is the *t_now* bank logic. It is based on the radar gimbal azimuth to the CM, the range from the CMMCA to the CM, the angle between the heading of the CMMCA and the heading of the CM (the heading crossing angle or HCA) and the turn direction of the CM (left turn, no turn, or right turn) (see Figure 14).

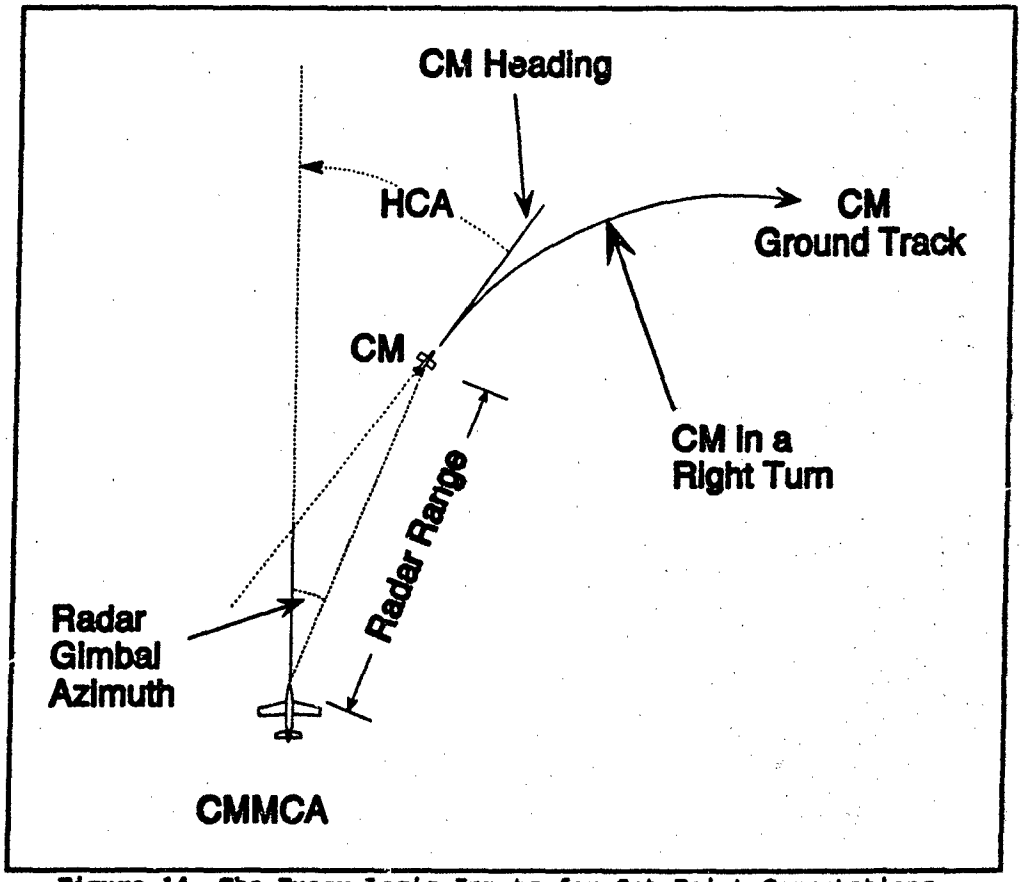

Figure 14 The Fuzzy Logic Inputs for Set Point Computations at Time t_now

The t_now bank value is most concerned with whether or not the range or radar gimbal azimuth is approaching a limit and getting worse. If so, the t_now logic demands a large bank angle to correct the condition. If no limits are near, or are close but the CMMCA is already correcting, then the t_now logic generally requires little bank.

The second part of the bank angle logic uses the future knowledge of the CM flight path. This is the *t_next* bank logic. The required bank angle is based on the HCA from the CMMCA's *present* heading to the

CM's future heading, and the radar gimbal azimuth from the CMMCA's present position to the CM's future position (see Figure 15).

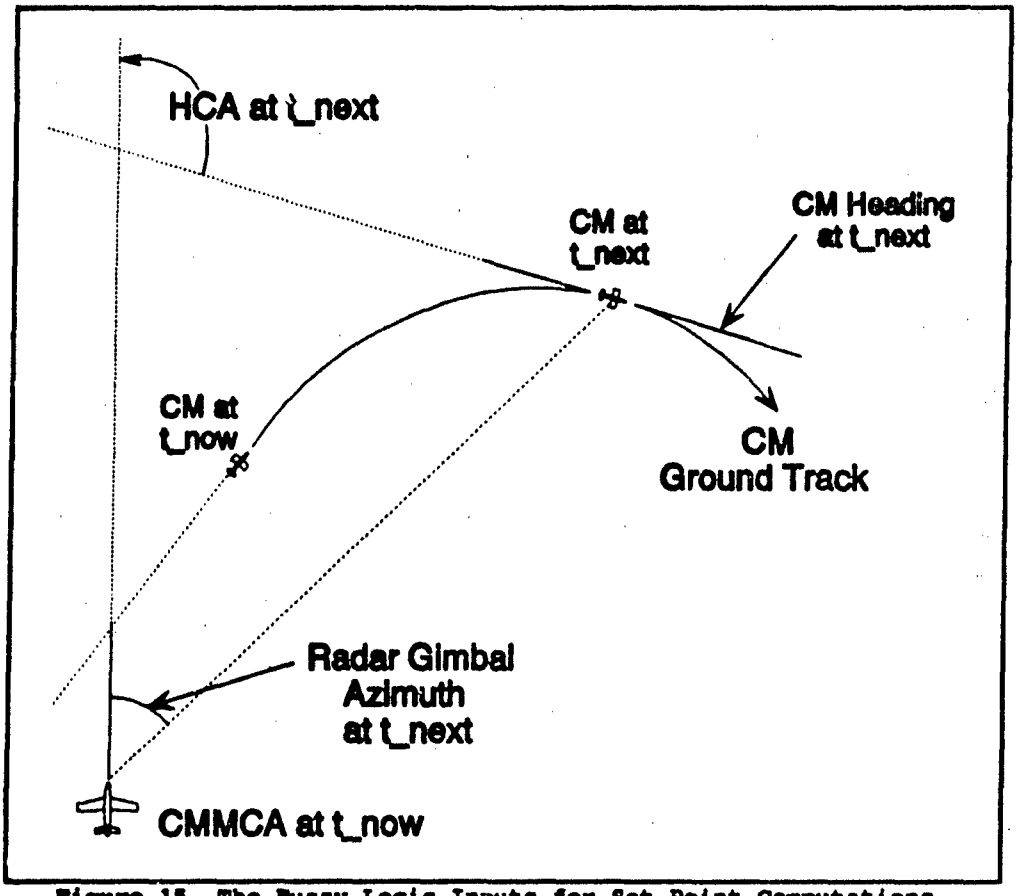

Figure 15 The Fuzzy Logic Inputs for Set Point Computations at Time t_next

The t_next logic works significantly differently than the t_now logic. It is biased towards eliminating future problems by building turning room early. This is done by the t_next logic generating CMMCA turns away from the CM in an attempt to create an offset. This is designed to avoid both losing radar contact from delayed CMMCA turns and decreasing range due to cutting off the inside of the CM's turn.

Fuzzy logic was then used one last time to combine the required bank angles derived from the present and future calculated values. The fuzzy logic that combined the t_now and t_next results was essentially a balancing act. If the limits were currently being approached, then the t_now required bank was the net result. On the other hand, if the

limits were currently not close, then priority was given to the t_next results. Some additional decisions were made to prevent the CMMCA from building too much offset by continually turning away from the CM.

An important difference exists between this thesis and the works of Garton and Hachman. They specified a nominal range (either eight or ten miles) and a nominal radar gimbal azimuth (zero degrees). Any deviation from the nominal resulted in the CMMCA applying control inputs to correct back to that position. Here, the entire region from approximately six to fourteen miles in range (leaving a one mile buffer) is considered to be completely acceptable. It is only as range reaches its limit that large controls are generated. Otherwise, only enough control is demanded to ensure the range stays within limits. The azimuth control is equivalent. Anywhere within roughly ±55 degrees is acceptable. Only as azimuth approaches the limits of ±60 degrees, are large bank angles demanded to return the CMMCA to within limits (Figure 16).

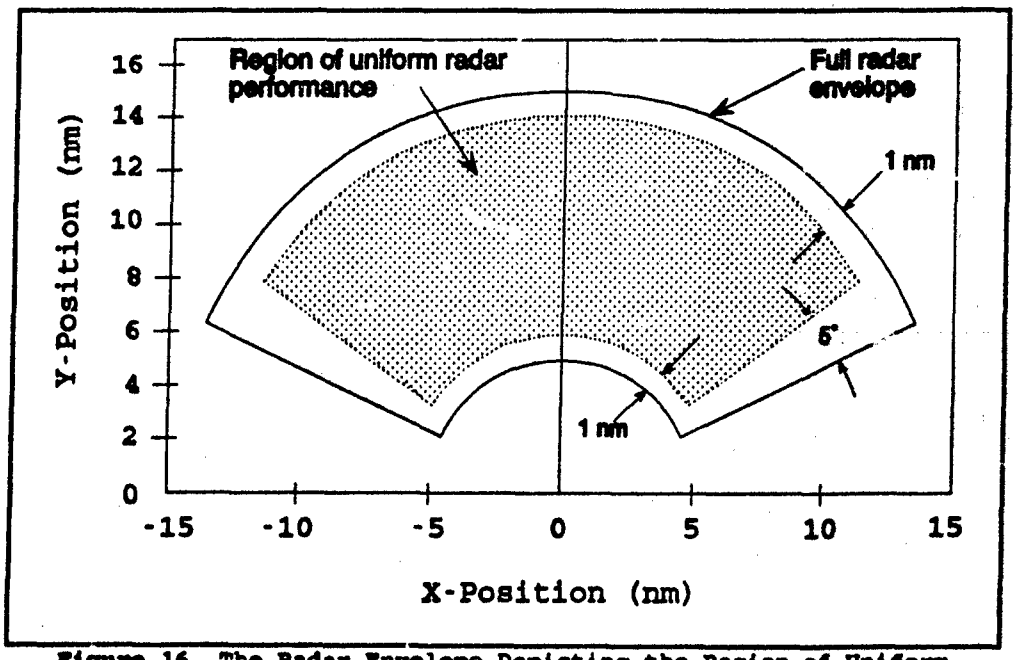

Figure 16 The Radar Envelope Depicting the Region of Uniform Performance

3.3 Computer Resources

3.3.1 Hardware and Operating Systems. Two different platforms were used in the development of the simulation. Much of the initial work was done on a Sun SPARCStation 2 running SunOS 4.1. The majority of the work was performed on an IBM compatible personal computer (PC). The PC used an Intel 80386 processor running at 25 Mhz, equipped with a math coprocessor chip, eight megabytes (MB) of RAM, a 65 MB hard disk and a VGA graphics card with 0.5 MB video memory. The PC was running MS-DOS 5.0 and Microsoft Windows 3.1.

3.3.2 Application Software. The entire simulation development effort was done in the Matlab environment. Matlab is a product of The MathWorks, Inc. of South Natick, Massachusetts. The large computer version (e.g. Sun Workstations and VAX's) is Pro-Matlab, while the IBM PC version is PC-Matlab.

Matlab "integrates numerical analysis, matrix computation, signal processing, and graphics..." and includes "...programmable macros, IEEE arithmetic, a fast interpreter, and many analytic commands (16:3)." It is driven from the command line, making it fully interactive, yet it has a significant programming capability in a matrix oriented, high level language.

Two of its most important features are its extensibility with M-files and its toolboxes. M-files are ASCII files of Matlab commands that are executed either sequentially in batch mode, or else add new, user defined functions to Matlab. The toolboxes are Matlab supplied function libraries in the form of M-files.

The Matlab toolbox Simulink is a way to simulate dynamic systems. It allows the user to build a simulation by adding and connecting blocks in a graphical environment. Simulink comes with dozens of predefined blocks and easily incorporates new functionality by incorporating M-files. Its interactive nature allows for real time debugging, as well as providing any desired output for examination or additional debugging.

Simulink is compiled at run time, and runs nearly as quickly as an equivalent FORTRAN program (17:3-11).

A number of M-files were written during the course of this thesis. The files in Appendix C were used prior to a simulation run. The first four M-files convert CM flight paths into the format needed during the simulations. The last five M-files are used to set up the global constants, find the Kalman gains used to compute the optimal controls, and generally to prepare everything for the simulation. The M-files of Appendix D were used during the simulation runs. They all appear in the overall block diagrams of Figure 18, Figure 19, and Figure 20 discussed later in this chapter.

The following versions of Matlab and its toolboxes were used:

PC-Matlab Version 3.5a (15)
Pro-Matlab Version 3.45 (16)
Control System Toolbox Version 3.03 (4)
Spline Toolbox Version 1.02 (18)
Simulink Version 1.2.1 (17)

3.4 Simulation Inputs and Parameters

The simulation is driven almost entirely by the capabilities of the CMMCA and by the flight profile of the CM. The CMMCA's parameters are perhaps the most important, yet the hardest to quantify exactly. As previously discussed, this thesis took the approach that the exact values were not required, merely reasonable values providing performance closely matching that of the CMMCA.

On the other hand, exact CM data was not needed because the CM performance was not truly modelled at all. The CM capabilities were embedded within its flight profiles, which were assumed to be objective fact. Based on the CM position as a function of time, various values such as heading and acceleration could easily be calculated.

3.4.1 CMMCA Parameters. As in all design problems, there are a large number of constants to be chosen, and it is seldom clear a priori what values to choose. A common method is to choose values based on experience and guesswork, and then adjust the values until the response

closely matches expectations. This method was adopted to set many parameter values used throughout the simulation. These included:

1. the maximum roll acceleration ($\ddot{\phi}$),
2. the roll coupling constant (ω),
3. the various times used (sample time, look ahead time and FTUT), and
4. the value of the cost functional coefficients found in the Q and R matrices of Chapter 2, equation 3.

A number of parameters were dealt with by simplifying assumptions as discussed at the end of Chapter 1. These included:

1. no sideslip (CMMCA in fully coordinated flight at all times),
2. CMMCA velocity constant at 400 kts,
3. uniform radar performance throughout entire region of coverage, and
4. CM initially at the center of the CMMCA's radar coverage.

Additionally, some parameter values were used essentially unchanged from Hachman's thesis. These parameters included the maximum bank angle and the radar coverage limits.

The initial value of the maximum roll acceleration was estimated from the roll performance of aircraft similar to the CMMCA. This led to a maximum value of roughly 8 deg/sec^2. Values for ω, Q and R were set initially at $\omega = 0.1$, Q and R as shown above in section 3.2.4 (2).

With these values set, a number of trial simulations were performed that forced the CMMCA to make a series of turns. Examination of the bank angle rates and accelerations was followed by a modification of the coupling constant to $\omega = 0.2$; Q and R were left unchanged. Through this trial and error procedure, the simulated turn performance closely matched that of the real aircraft. These values were then fixed for the remainder of this thesis.

The several time parameters were chosen for rather pragmatic reasons. The sample time of one second was selected because it seemed small enough to capture any significant details, yet not so small as to vastly increase computer time. The shortest and longest flight profile times were 442 and 1850 seconds, hence a sample time of one second was a good compromise value.

The value of FTUT was originally chosen to be approximately an order of magnitude greater than the sample time. Because the simulation

worked well with the chosen values of sample time and FTUT, there was subsequently never any reason to change these values.

Look ahead time, however, is subject to much greater variability. The values that were finally chosen for consideration were: 60, 90 and 120 seconds; simulations using all three times were run. Look ahead time is discussed in more detail in Chapters 4, 5, and 6.

3.4.2 CM Parameters. The only CM data known at the outset is the initial position and ground track. The rough position data is converted first by decomposing it into a series of circular and straight segments. Because the position data is sparse, these segments would, in a sense, be *best fit* curves. For this thesis effort, the CM data was taken as if raw data had already been fitted to a series of circular and straight segments.

The position of the CM was computed at intervals of one second (the sample time) along each segment for the entire time of flight. The position data was used to generate a matrix of CM headings, positions and directions of turn at intervals corresponding to FTUT (every ten seconds) for the entire time of flight. This matrix was used by the fuzzy logic to compute the next set point. Although this could have been done at each set point calculation, pre-computing this matrix sped up the simulation.

The position data was also run through a spline generating Matlab M-file and differentiated twice to produce the (inertial) accelerations of the CM. The purpose of generating splines was because of the sparseness of the actual CM position data. It is extremely simple to interpolate along a spline; also the derivatives thereby produced will be much smoother than from the raw data.

Circular segments were characterized by four parameters, the inertial position and velocity at the start of the turn, the direction of turn and the numbers of degrees to turn. Straight segments were characterized by three parameters, the initial position, the heading, and the length of the segment.

The transformation to circular and straight segments was simplified by the assumptions that the CM was flying at a constant velocity and that, when turning, it did so at a constant bank angle. As reported by Hachman, the CM used 20 degrees of bank in his profile four (7:35). This thesis examined Hachman's profile four using both 20 and 30 degrees of bank.

3.5 Scenarios

All scenarios began with the CMMCA 10 nm in trail of the CM and displaced 1000 feet to the West (left). Because this nominal range is a slant range, the CMMCA has an actual ground range of

$$\sqrt{(10*6076)^2 - (-29000-(-1000))^2) - (0-1000)^2} = 53,915 \text{ ft}$$

With no loss of generality, the CMMCA is initially placed at the origin of the inertial coordinate axis system and the CM due north at coordinates X = 53915, Y = 1000. Hence both vehicles also have initial headings of 360 (or 000), and initial X and Y velocities of $\dot{X}$ = 675.11 ft/sec and $\dot{Y}$ = 0. The CMMCA's Z coordinate is fixed at -29,000, the CM's Z coordinate at -1,000, and both their vertical velocities are $\dot{Z}$ = 0.

This thesis examined the performance of the CMMCA over three different scenarios. They corresponded to Hachman's profiles one, three, and four and are repeated in Figure 17. To prevent confusion between nomenclature, they will be referred to as profiles A, B, and C.

Profile A was essentially a quick response, tuning and experimental scenario. The simulation and fuzzy logic were developed and tested first against this profile because of its simplicity and because of the much smaller simulation time (442 vs 1850 seconds for profile A vs profile C); consequently any debugging involved far less output to examine. Since profile A consisted of only one turn, it was ideal for developing the most fundamental portion of the tracking solution – a single turn by the CM.

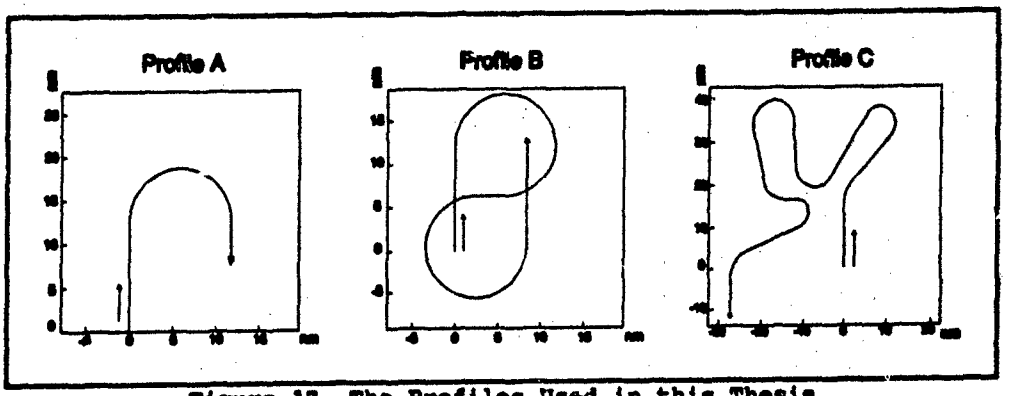

Figure 17 The Profiles Used in this Thesis

Profile B contained two large turns, and was thus a useful means of testing the ability of the CMMCA to transition from one CM maneuver to the next. This profile was used to further tune the fuzzy logic to ensure it could position the CMMCA properly between maneuvers.

Profile C was important for two reasons. It represented a challenging real world CM profile. Also, it is important to test the results of the simulation development on a profile that had not been used to construct the simulation. In essence, profile C provided an independent verification of the results of the CMMCA tracking solution. Because the value of look ahead time was not fixed, runs that looked promising were performed at all values of look ahead time (60, 90 and 120 seconds).

3.6 Simulation Outputs

Simulink allows output of any value during or after the simulation. As an aid to debugging, the output included various intermediate fuzzy logic calculations, radar range and radar gimbal angles, the states of the CMMCA and CM, the bank, pitch and heading of the CMMCA, the commanded and actual controls (roll acceleration and inertial components of the X, Y and Z accelerations) and the simulation time. Also output, as required for debugging on different occasions, were various results produced internal to the several M-files as they computed set points, limited controls, converted body to inertial coordinates, etc.

To determine the success of the simulation, the radar range and gimbal angles were needed. To provide plots of the CMMCA and CM, and to show their relative position, the only additional information needed was the inertial positions of the two vehicles, which came directly from their state vectors.

The form of the output from the simulation was easy to interpret. Simulink by default generated an output value at each sample time. Since most outputs were vectors, the full output over the simulation was a matrix with the number of rows equal to run time and the number of columns equaling the size of the output vector. For example, the CMMCA's state vector output was a 442 by 8 matrix for profile A, or 28,288 bytes (numbers used 8 bytes of storage each).

3.7 The Simulation

A block diagram of the simulation is shown in Figure 18. It can be broken into five main sections:

1. the CM state-space,
2. the CMMCA state-space,
3. the set point calculation including fuzzy logic,
4. the controls to be applied, and
5. the bank angle acceleration limiter.

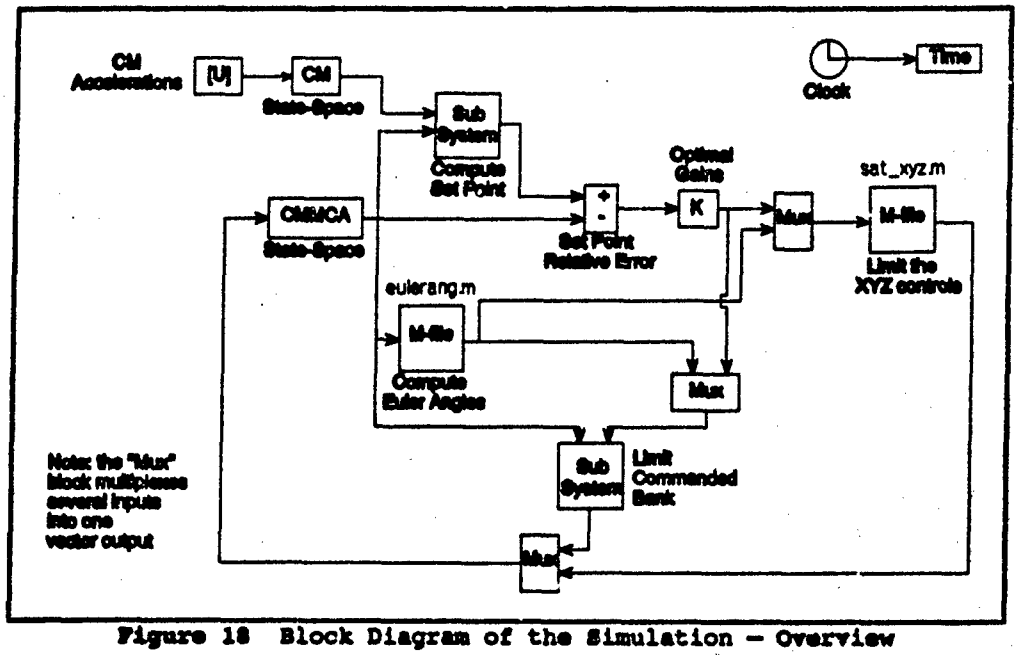

Figure 18 Block Diagram of the Simulation – Overview

The CM state-space calculations have for their input the accelerations from the profile ground track and are discussed in section 3.4.2. These accelerations are fed into the state-space equations and integrated at each sample time.

The CMMCA state-space calculations are similar to that of the CM, except that the accelerations come from the other blocks of the system, rather than externally from the ground track information. The value of the state variables at each sample time are also used to compute the euler angles, that is, the heading, pitch and bank angle. The heading (φ) is derived from the projection of the velocity onto the XY plane. The pitch (θ) is simply the angle derived from the velocity component along the Z axis. The bank angle comes directly from the state vector.

The set point calculation is shown in more detail in Figure 19. It takes the CM and CMMCA states and the CMMCA euler angles as inputs, producing the next set point as output. The first block represents an M-file that computes the radar range and radar gimbal angles from the CMMCA to the CM. If the simulation time has not yet advanced to FTUT, then the current set point is extracted from memory and the fuzzy logic is skipped.

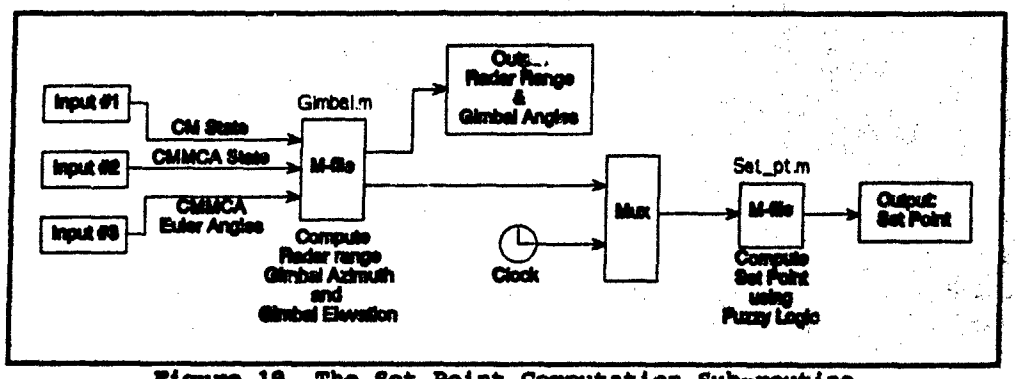

Figure 19 The Set Point Computation Sub-routine

If simulation time has reached FTUT, then the states, euler angles and radar values are fed into the fuzzy logic M-file. The fuzzy logic has already been described in section 3.2.6. The output of the fuzzy logic is the inertial XYZ set point; this is augmented with the current

CMMCA bank angle, X, Y and Z velocities and bank angle rate to provide the output in state space form.

With the current state and set point of the CMMCA available, the simulation next computes the controls needed to drive the CMMCA to the set point. This is carried out in the blocks in the upper right corner of Figure 18. The relative error between the present CMMCA state and the set point is computed and multiplied by the control gain matrix K derived from the solution to the Riccati equation.

These controls are the XYZ inertial controls. They are combined with the euler angles to transform the controls to accelerations in the body axes. Every attempt is made to preserve the direction of the commanded acceleration. Because the CMMCA is at a constant speed, the body X-axis acceleration is set to zero. Similarly, because no sideslip is allowed, the body Y-axis acceleration is set to zero.

Hence the only accelerations allowed are along the body Z-axis and the bank angle acceleration. The bank angle will be dealt with below. Since the CMMCA currently moves at constant altitude, the body Z-axis acceleration must just balance the force of gravity. It turns out, then that the constant altitude, constant airspeed and no sideslip requirements constrain all accelerations except that of the bank angle.

The bank angle acceleration calculations are shown in full detail in Figure 20. The needed inputs are the unlimited X, Y and Z inertial accelerations, the euler angles, the current bank angle and the bank angle rate. As described above, the inertial X, Y, and Z accelerations are converted to the body axes and then projected onto the YZ body axis plane. The bank angle producing this direction of acceleration is then easily found by setting the CMMCA's wings perpendicular to this projected acceleration.

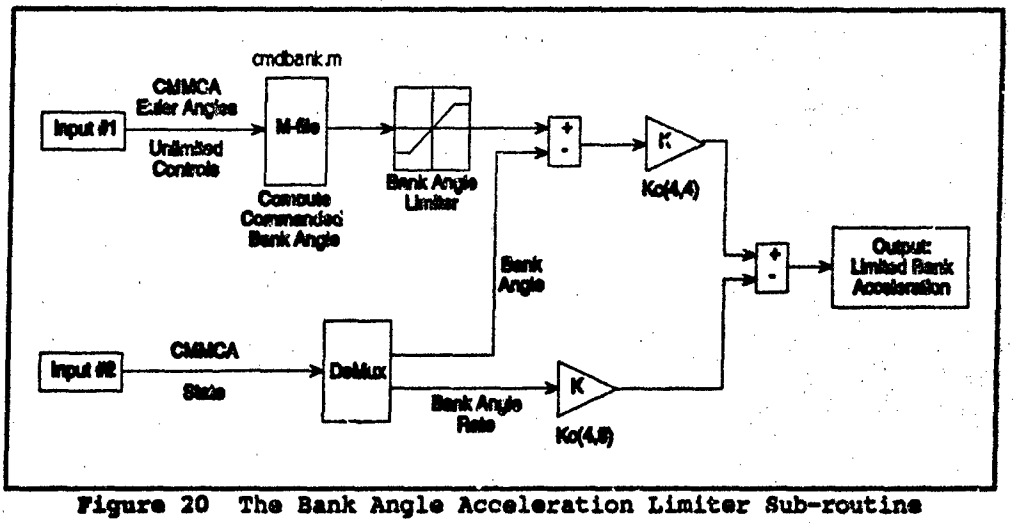

Figure 20 The Bank Angle Acceleration Limiter Sub-routine

This commanded bank angle is then limited to the CMMCA's maximum bank angle. The result of Figure 2C is

$$\dot{\phi}_{commanded} = (\phi_{limited} - \phi_{actual}) * K_c(4,4) - \dot{\phi}_{actual} * K_c(4,8)$$

which is just the Kalman gain applied to the bank angle and bank angle rate. This commanded bank angle acceleration is then limited to the maximum value of eight deg/sec^2.

IV. Controller Development

4.1 Introduction

This thesis successfully built a simulation using optimal control theory and fuzzy logic to solve the CMMCA tracking problem. The simulation was built and verified using an incremental approach. This chapter describes how the simulation was constructed, the problems discovered and the fixes introduced during its development.

4.2 Simulation Development

The simulation was developed in stages. The first stage was to build a means of generating CM flight paths. The second stage was to develop an *ideal* optimal controller. This controller was ideal in the sense that it drove an infinitely maneuverable CMMCA to one mile in trail. The next step was to incorporate the actual limitations of the CMMCA into the simulation. Finally, the one mile trail position was changed to the actual radar envelope of the CMMCA.

4.2.1 The CM Flight Path Generator. This has been partially discussed in section 3.4.2. The flight path generator was written as a general purpose Matlab routine using several M-files. The CM bank angle, and consequently its turn radius, is hard-wired into the flight path generator. Currently it is set to 20 degrees. The output is written to a disk file for permanent storage. All of these computations are performed *off-line* to speed up the simulation.

The input is a flat data file consisting of an n × 2 array. Each of the n rows corresponds to either a straight or turning flight path segment. The first column is that segment's turn direction: -1 for a left turn, +1 for a right turn and 0 for no turn. The second column is a duration – the number of degrees to turn or the distance (in feet) to go straight.

After reading this data file, the generator takes each segment in turn, creating an array of points along that segment at one second

intervals. Then a spline is fitted to these points and differentiated twice to provide an array of CM accelerations. After all segments are complete, they are combined into the array "U" shown as the CM accelerations in the upper left of Figure 18, Chapter 3. This array is also used to generate a second array of CM headings, X and Y positions, and direction of bank at one second intervals.

4.2.2 The "Ideal" Optimal Controller. This is implemented as shown in Figure 21, a simplified version of the ultimate controller of Figure 18. At this stage of the simulation development, the CMMCA has no limits on its performance: it rolls instantaneously to any bank, changes to any speed in zero time, and is capable of generating infinite accelerations. The CMMCA's position has been set to one nautical mile (ground range, not slant range) in trail of the CM.

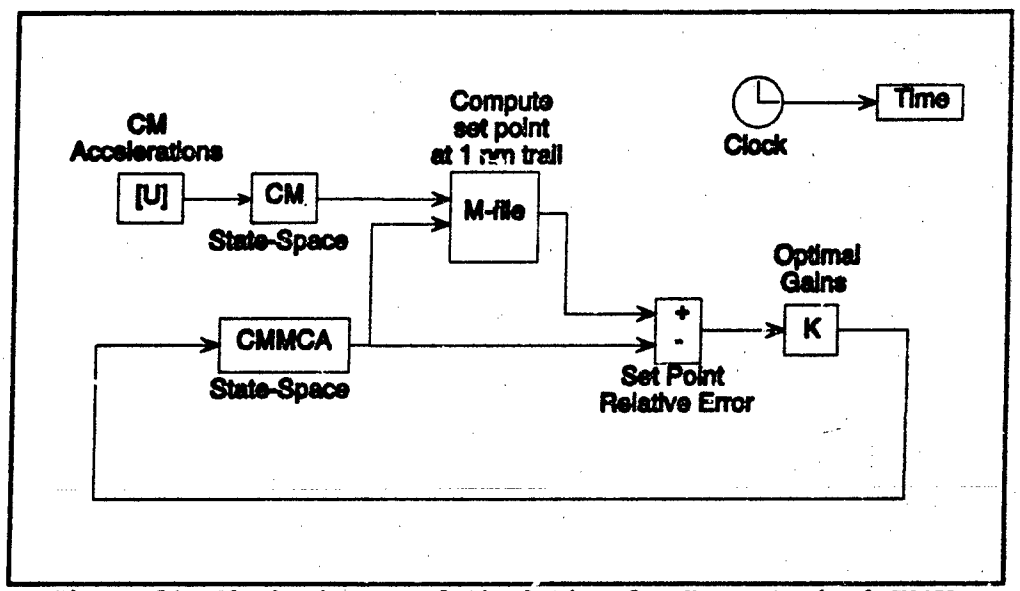

Figure 21 Block Diagram of Simulation for Unconstrained CMMCA Maneuvering

Because of the unlimited controls, the CMMCA, should be able to exactly track the CM. At each time step, the set point is computed one mile dead astern the CM. Then the optimal controls are applied to the CMMCA to drive it there in that one second time interval. This means there is no residual error; at each sample time control calculation, the CMMCA starts precisely from where it is supposed to be.

This stage of development had two major purposes: 1) to become completely familiar with Simulink and all of its complexities, intricacies and idiosyncracies; 2) to develop a working, albeit vastly simplified (and still incomplete) simulation. One of the many problems with this stage, of course, is the that the lack of limits leads to CMMCA accelerations of many hundreds of "g's," and speed changes of thousands of knots. This is dealt with at the next stage.

The reason for one mile trail as opposed to the nominal center of radar coverage at ten miles is for simplicity. A position ten miles in trail would result in significant *crack-the-whip* problems as the CM turned. The intent was not tracking to maintain radar coverage, but simply to control the CMMCA.

This portion of the simulation building went relatively smoothly. The CMMCA was able to track the CM on any profile under consideration. Achieving this step provided the ability to fall back to a working simulation if problems were encountered later in any of the stages.

4.2.3 The Optimal Controller with Limits. Incorporating the physical limitations of the CMMCA into the simulation resulted in a simulation close to that of Figure 18, Chapter 3. However, the set point is the same as in the previous section, that is, one mile trail. In this step, the CMMCA is limited in bank angle, roll rate, "g" capability and speed change. Hence, the CMMCA cannot generally reach the next set point. Nevertheless, the CM profile should allow the CMMCA to track it well from the one mile trail position.

Tracking was not directly a major goal at this stage. The real goal was to properly code the CMMCA physical restrictions into the simulation. Tracking was merely feedback to verify the restrictions were implemented correctly.

The first serious difficulty was encountered at this point. The idea was to compute the commanded CMMCA inertial accelerations, and then to limit this acceleration based on the CMMCA's capabilities while maintaining the direction of the original, unlimited acceleration. This

led to conceptual problems as to exactly what that direction meant and how to properly limit the acceleration.

As mentioned in the previous chapter, the inertial acceleration was transformed into the body axis coordinate system. Then the X component was limited independently, since it represented a change of the CMMCA's speed. The Y and Z components together represented the acceleration of the CMMCA due to lift from the wings. It was unclear, however, whether this also should include gravity.

Eventually, it became clear that the YZ body acceleration was the direction to maintain and did include gravity. In other words, the CMMCA's "g" acceleration, sideslip acceleration and the acceleration due to gravity should add up to an acceleration in the same direction as the commanded YZ body acceleration.

This was often not possible, however. Suppose the CMMCA was in thirty degrees of left bank and the commanded acceleration was to the CMMCA's right. No combination of allowable sideslip and "g" would allow the CMMCA to go right while in left bank. The allowable sideslip was in the range of 1/4 to 1/10 of a "g," moreover, the desire was for no sideslip. Therefore, the decision was made to restrict the CMMCA to zero sideslip. Its contribution to CMMCA capabilities was minuscule, it would never actually be used by the aircrew, and it vastly increased the complexity of the control limiting algorithm.

With sideslip limited to zero, and speed controlled independently, the only acceleration left was along the Z-body axis. Maintaining the direction of the commanded acceleration was then a simple matter of finding the "g's" that, together with gravity, made their sum as close yas possible to the original commanded acceleration.

The heart of the limiting process, then, was not really the X, Y or Z accelerations, but rather the bank angle accelerations. The CMMCA can really only accelerate in the direction in which it's banked. Thus it must be forced to bank in the required direction, as determined by

the direction of the commanded acceleration. The bank angle acceleration calculation was shown in Figure 20, Chapter 3.

This calculation is really just multiplying the CMMCA states by the optimal gains, extracting only the $\ddot{\phi}$ (fourth) component of the control, and then limiting the resulting CMMCA bank angle acceleration command to match the CMMCA's capability. The $-\phi \cdot K_c(4,8)$ portion of the calculation (lower half of the figure) serves to decrease the acceleration to zero as the roll rate grows. The upper half of the figure corresponds to decreasing the acceleration to zero as the bank angle itself approaches maximum.

This is one important area where this thesis differs markedly from the prior three. Here the bank angle is automatically kept within limits; the structural limits are never exceeded (there is actually a slight over-bank of up to one degree due to aircraft dynamics). Thus only the radar limits can be exceeded with this method.

To verify the proper working of the limiter algorithm, the CMMCA was simulated over several flight paths. The early trials were with short profiles, and as confidence increased, longer profiles were used. The current algorithm is apparently correct, since the CMMCA has the ability to track the one mile trail position quite well over a fairly convoluted profile. It will not track perfectly due to its designed limitations. The cause of most of the error between the CMMCA's present position and the set point is because of the delay imposed for the CMMCA to roll to the needed bank angle and begin turning.

4.2.4 The Radar Envelope Set Point with Fuzzy Logic. At this stage, the CMMCA does not yet accurately track the CM, since no account is made of the information supplied by the radar or the knowledge of the CM's future ground track. The controller needs to dynamically compute and alter the set point based on the radar and the CM's flight path. This is the final stage of simulation development wherein fuzzy logic was used to determine set points.

The transition from the previous simulations to this last stage involved a fundamental conceptual shift. The prior work involved determining set points relative to the CM – one mile trail of the CM. At this stage the shift is towards making the set point relative to the CMMCA itself. Section 3.2.6 discussed set point determination in detail; the next section will discuss the fuzzy logic that determines the set point at intervals of FTUT.

4.3 Development of the Fuzzy Logic

4.3.1 Introduction. The fuzzy logic was discussed at length in sections 3.2.6. The nature of fuzzy logic requires that expert knowledge, opinion and intuition be a fundamental portion of the set point logic. The logic was used to drive the CMMCA to successive set points, each one chosen to maintain current radar coverage and to ensure that radar coverage would be possible in the future.

The prime means of ensuring future radar coverage was to recognize approaching problems and to avoid them by having the CMMCA maneuver early. Recognizing problems early meant examining where the CM will be at the look ahead time (t_{ahead}), as well as what it's doing then. Difficulties arose any time the CM went into a long turn (exceeding roughly 90 degrees). As discussed in Chapter 1, when facing a large CM turn, the CMMCA is hard pressed to maintain radar coverage.

The solution appeared obvious (from the pilot's perspective): turn the CMMCA slightly away from the CM's upcoming turn to build turning room, turn back into the CM to prevent it drifting out the side of the radar coverage, then turn back inside the CM's turn circle (see the plots in Appendix A for examples). The initial offset allows the CMMCA to delay its eventual large turn. This delay prevents the CMMCA from drastically decreasing the range, because otherwise it would be forced to turn much earlier, closing the distance significantly.

Another important, yet related portion of the logic occurs after finishing a maneuver and before beginning the next. Instead of the CMMCA simply returning to a trail position, it is better for it to

displace off to one side. The side chosen should be the side opposite the next turn direction, thus automatically building in turning room.

The fuzzy logic was implemented in five logic tables. Three tables computed the contribution to the CMMCA's bank angle at t_now, one table at t_next, and the last table combined the t_now and t_next values. Each table was purposely designed to stand alone; that is, only the inputs for that table were taken into consideration. Although it seems artificial to separate out and ignore other factors, the alternative is actually to create one extremely large and complex table with six or eight inputs and as many logical operations. Doing this would defeat the entire purpose of using fuzzy logic: the intuitiveness and simplicity.

Furthermore, the different inputs were all considered by using nested logic. For example, four factors were used, two each in two different tables, to determine the two inputs to the t_now bank value. Then they were in turn used as the inputs to a third table to compute the t_now bank command. This fashion of modularizing the fuzzy logic was similarly used to combine the t_now and t_next values.

The fuzzy logic tables that follow in this chapter represent the final results at the conclusion of the experimental process. A discussion of the evolution of the table entries is located in section 5.2.

4.3.2 The t_now Fuzzy Logic. Although creating an offset was important to prevent future CMMCA tracking problems, it was equally important to preclude loss of radar coverage in the next few seconds. The t_now bank command logic was crafted to avert imminent loss of radar coverage in the extreme case, and to force the CMMCA to follow the CK with more moderate controls in other cases. This logic (Table 6) was synthesized from the outputs of two fuzzy logic tables (Table 5).

Table 5a uses the current azimuth to the CM and the heading crossing angle (HCA). Table 5b uses the range to the CM, the CM's turn direction and the sign of the radar gimbal azimuth (±1,0). Both tables

are based on the variables displayed in Figure 14 and Figure 15 and discussed in section 3.2.6.

Table 5 Logic Tables for t_now Bank Using Azimuth and HCA, and Range and CM Turn Direction

Azimuth	HCA	azi_hca	row #	Range	CM turn direction	rng_dir
	LP	LP	1			
	SP	LP	2		POS	LP
LP	ZR	LP	3	LP	ZERO	LP*D
	SN	SP	4		NEG	LN
	LN	SP	5			
	LP	SP	6			
	SP	SP	7		POS	SP
SP	ZR	SP	8	SP	ZERO	SP*D
	SN	ZR	9		NEG	SN
	LN	SN	10			
	LP	SP	11			
	SP	SP	12		POS	ZR
ZR	ZR	ZR	13	ZR	ZERO	ZR
	SN	SN	14		NEG	ZR
	LN	SN	15			
	LP	SP	16			
	SP	ZR	17		POS	SN
SN	ZR	SN	18	SN	ZERO	SN*D
	SN	SN	19		NEG	SP
	LN	SN	20			
	LP	SN	21			
	SP	[illegible]	22		POS	LN
LN	ZR	[illegible]	23	LN	ZERO	LN*D
	SN	LN	24		NEG	LP
	LN	LN	25	Note: D is the sign of the azimuth to the CM (±1,0)		

The logic for the azimuth/HCA table (Table 5a) is straightforward. If the CM is nearly at the edge of the radar azimuth limit and not correcting back, then the CMMCA should make a hard turn towards the CM. If the CM is nearly off the side of the scope, but is correcting back, then the CMMCA need only make an easy turn towards the CM. If the CM is off to the side but not close to the edge of the radar envelope,

then the CMMCA should essentially make an easy turn in the direction that the CM is heading.

For example, if the CM is to the right and going further right (lines 6 & 7 of Table 5a), then the CMMCA should make an easy right turn to hold the CM at the same azimuth. If instead the CM had been moving back to the left (lines 9 or 10), then the CMMCA should maintain heading or make an easy left turn. If the CM is at small azimuths (lines 11-15), then the CMMCA should make an easy turn in the direction of the CM's heading.

Table 5b implements the fuzzy logic to control the CMMCA's turn using the radar range to the CM, the CM's turn direction, and the azimuth to the CM. The decision logic is simple: if the CMMCA is too far behind the CM, it needs to decrease the range by cutting across the CM's turning circle. If the range is too small, the CMMCA needs to turn away from the CM to increase the range. When the CM is turning, the direction for the CMMCA to turn is either in the same or opposite direction as the CM. However, when the CM is not turning, then the sign of the azimuth to the CM is used to determine which direction is away from the CM (the factor of "D" in the table).

Table 6 consists of the logic used to integrate the two bank angle values (scaled) computed by Table 5. This integration is based on passing through unchanged any large bank angle command (LN or LP) from either component. This is biased slightly in favor of the azimuth/HCA components versus the range component. When neither component is large, the output of the corresponding rule is, in some sense, the average of the two components.

4.3.3 The t_next Fuzzy Logic. The fuzzy logic to determine the CMMCA turn at the look ahead time is more complicated. It is based on building turning room early by having the CMMCA turn away from the CM (see Table 7). None of the outputs from this table are large. This is to ensure that while a slight turn away from the CM can occur, large turns will not happen, and that large t_now values will swamp t_next

Table 6 Logic Table for t_now Bank Combining Results from Azimuth/HCA and Range/CM Turn Direction

IF azi_hca is	AND	rng_dir is	THEN	now_bank is
LP		any value		LP
SP		LP		LP
		SP		SP
		ZR		SP
		SN		ZR
		LN		SN
ZR		LP		LP
		SP		SP
		ZR		ZR
		SN		SN
		LN		LN
SN		LP		SP
		SP		ZR
		ZR		SN
		SN		SN
		LN		LN
LN		any value		LN

values when those results are combined. Also, it solved an early problem wherein the CMMCA continually turned away from the CM, never turning back towards it.

4.3.4 The Net Bank Fuzzy Logic. At this point, there is one bank command value usually forcing the CMMCA to turn towards the CM in order to track it (the t_now value), and another forcing the CMMCA to offset by turning away from the CM. Balancing the near term and future requirements was therefore a critical task. The means of accomplishing this often contradictory task were discussed in section 3.2.6 in general terms. The final form of the logic is shown in Table 8.

The first and last row of the table correspond to the situation where the CMMCA must turn immediately or lose radar contact. The output for all such cases is made independent of any t_next consideration, and is a hard turn towards the CM to preserve radar coverage.

Table 7 Logic Table for t_next Bank Using Azimuth and HCA at the Look Ahead Time

IF next_azi is	AND	next_HCA is	THEN next_bank is
LP		LP	SN
		SP	SN
		ZR	SP
		SN	SP
		LN	SP
SP		LP	SN
		SP	ZR
		ZR	ZR
		SN	SP
		LN	SP
ZR		LP	SN
		SP	SN
		ZR	ZR
		SN	SP
		LN	SP
SN		LP	SN
		SP	SN
		ZR	ZR
		SN	ZR
		LN	SP
LN		LP	LN
		SP	LN
		ZR	SN
		SN	SP
		LN	LP

All rows other than the first or last correspond to t_now commands for a small or zero turn; the CMMCA only needs a small bank to track the CM and the need to turn is not urgent. Here is where the balancing of present and future commands occurs. The majority of weight is given to the t_next input, since the CMMCA is not in any immediate danger of losing radar coverage. Negative t_now values (left turns) result in an output equal in size but opposite in direction as for the positive case.

4.3.5 The Fuzzy Set Membership Functions. While the decisions implemented in the fuzzy logic obviously drove the commanded bank for

Table 8 Logic Table Combining t_now and t_next Fuzzy Logic Inputs

IF t now bank is	AND t next bank is	THEN net bank is
LP	any value	LP
SP	LP	LP
SP	SP	SP
SP	ZR	ZR
SP	SN	ZR
SP	LN	SN
ZR	LP	SP
ZR	SP	SP
ZR	ZR	ZR
ZR	SN	SN
ZR	LN	SN
SN	LP	SP
SN	SP	ZR
SN	ZR	ZR
SN	SN	SN
SN	LN	SN
LN	any value	LN

the CMMCA, it turned out that the membership functions were also important. The first half of choosing a membership function was to determine how the raw, analog values should map into the scaled range [-5,5]. The second half of the membership function was the shape of the function itself. The shape and spacing of the membership functions suggested in the literature was shown in Figure 7, Chapter 2.

The mapping applied to five of the factors used in the fuzzy logic tables. They are summarized, along with their possible and usual ranges, in the first three columns of Table 9. It seemed reasonable to map the usual ranges shown in column three of Table 9 to the range [-5,5]. However, the performance of the CMMCA needed to improve at two places: as the CM approached approximately 30 to 45 degrees of azimuth at t_now, and during the time the CM was flying straight and level following a large turn.

The first performance problem was because the CMMCA offset away from the CM for turning room and consequently the current azimuth

Table 9 The Fuzzy Logic Factors and Their Ranges

Factor	Possible Range	Usual Range	Mapped Range
t_now azimuth (degrees)	[-180,180]	[-60,60]	[-60,60]
t_now HCA (degrees)	[-180,180]	[-180,180]	[-135,135]
t_now range (nm)	[0,infinity]	[5,15]	[5,15]
t_next azimuth (degrees)	[-180,180]	[-180,180]	[-60,60]
t_next HCA (degrees)	[-180,180]	[-180,180]	[-180,180]

approached the limit. The net bank logic gave too much weight to the t_next input and continued to offset the CMMCA until too late. At that point the CMMCA could not turn fast enough to keep from losing radar contact with the CM.

The second problem was at the end of the CM's maneuvering. The CMMCA would finish its turn and end with an HCA of approximately 25-35 degrees. Then it would continue to diverge from the CM's heading until the azimuth limits were approaching. This divergence also meant that the CMMCA was not driving towards a position from where it would be properly set up for the next maneuver.

The solution to both of these problems was accomplished by changing both the mappings to the scaled range and by re-defining the fuzzy set membership functions. The new mappings amounted to changing two factors from their usual ranges, the t_now HCA and t_next azimuth. Reducing the mapped range of t_now HCA reflected the need to react more to current HCA than was being done, while reducing the t_next azimuth range kept the CMMCA from turning too far during its offset maneuver.

Instead of changing the mappings and the set membership functions, the same results could have probably been accomplished by modifying the fuzzy logic rules. However, the former changes seemed the most expedient and straight forward, while retaining the intuitive nature of the fuzzy logic. Changing the logic rules would have required losing those advantages to force the same outcome.

the fuzzy logic. Changing the logic rules would have required losing those advantages to force the same outcome.

The final factor ranges are shown in the last column of Table 9. Any values that exceed these ranges (such as a t_next azimuth of 80 degrees) are simply truncated to ±5 during the mapping. The final fuzzy set membership functions are shown in Figure 22. The major change is a large expansion in the size of the set "ZR." This has a two-fold effect: t_now maneuvering is only performed as necessary, and the CMMCA terminates its offset steering after only a small turn away from the CM.

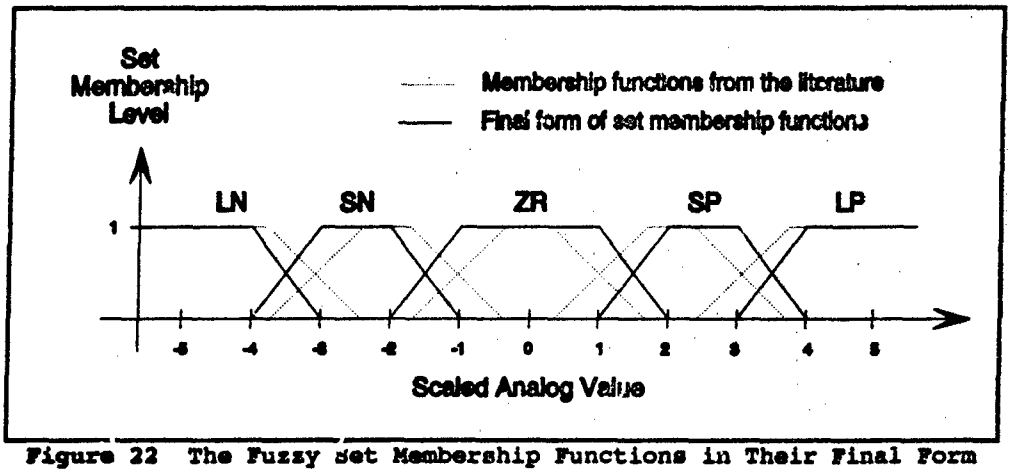

Figure 22 The Fuzzy Set Membership Functions in Their Final Form

V. Results

5.1 Introduction

This chapter presents the results obtained from fourteen simulation runs. The first section steps through the runs, with a discussion of what was varied to improve the simulation. The last section contains information on the simulation run times, and some ways to improve them.

5.2 Results

The results for simulation runs during the testing and tuning of the fuzzy logic (as discussed previously) can be seen in Table 10. The results in Table 10 are listed in the chronological order that the simulations were run, which represents an evolution of the set point fuzzy logic. Within each row of Table 10, only the look ahead time was changed, while from one row to the next, either the fuzzy logic rules or the set membership functions were changed. Appendix A contains plots of these simulation results. Appendix B contains a small, randomly selected sample of the output data produced during simulation Run #7c from time t = 1234 seconds to t = 1278 seconds.

5.2.1 Run #1, t_ahead = 90, 120. The results of Row #1 were not especially good. The problem arose early when the CMMCA offset to the left, turning too far and maintaining a diverging heading for too long. Both maximum range and maximum right azimuth were exceeded for long periods.

5.2.2 Run #2, t_ahead = 60, 90, 120. The radar range problem was aggressively attacked on two fronts: by rescaling the mapping from actual range to the scaled range [-5,5], and by changing the fuzzy logic for t_now commanded bank using radar range as an input.

The new range mapping meant that the range became fully LN or LP as the range reached the buffer region (greater than 14 nm or less than 6 nm). The new range/turn direction component logic gave more weight to

Table 10 Results for Trial Simulations Using Various Combinations of Fuzzy Logic and Look Ahead Times

Profile & CM bank	Limit (percent within)	Look Ahead Time 60	90	120	Row #	Change from Previous Row
A & 20°	range azimuth elevation		87.6 76.3 100.0	69.1 74.9 74.9	(1)	n/a
A & 20°	range azimuth elevation	100.0 83.5 100.0	100.0 84.4 100.0	100.0 85.8 100.0	(2)	map of radar range to [-5,5]; range logic
A & 20°	range azimuth elevation	100.0 84.2 100.0			(3)	set membership functions
A & 20°	range azimuth elevation	100.0 87.1 100.0			(4)	turn offset logic
B & 20°	range azimuth elevation	100.0 81.2 100.0			(5)	none
A & 20°	range azimuth elevation	100.0 100.0 100.0			(6)	turn offset logic; scaling;
C & 20°	range azimuth elevation	100.0 90.3 99.8	100.0 93.0 99.9	100.0 100.0 100.0	(7)	none
C & 30°	range azimuth elevation	100.0 92.0 100.0	84.9 79.0 95.4	82.7 78.6 97.2	(8)	none

radar range as it approached its limits than in the previous formulation. In particular, a range of either LN or LP now resulted in a large output bank from this logic component. The t_now and the net (t_now vs t_next) combining logic were also modified to give more weight to any t_now component or combined LN or LP bank angle. The improvement was immediate and obvious, as shown in Row #2. For all subsequent trials with the CM at 20 degrees of bank the range limits were never violated.

New HCA scaling was also used to keep the CMHCA from turning too far left and to turn back towards the CM when necessary. A ten percent improvement across all look ahead times was noted, although the new

range logic may have contributed as much to the azimuth improvement as the azimuth logic itself, by forcing the CMMCA to turn towards the CM as maximum range was approached.

A foreshadowing of future problems was also seen at the end of the simulation (see e.g., page 81, Appendix A). For all three look ahead times, the CMMCA ends the simulation with a large heading away from the CM and no attempt to correct back. This could potentially degrade the CMMCA performance if the CM were to make another turn.

However, since the CM flight path has ended, the heading divergence may simply be an *end effect*. A possible solution would be to artificially extend the CM's path in a straight line beyond its final position for a distance equal to the look ahead time. This would provide a virtual flight path for the CMMCA and bring it to a more reasonable final trajectory.

5.2.3 Run #3, t_ahead = 60. Row #3 shows the result of changing the set membership functions for the purpose of decreasing the CMMCA offset and improving the azimuth problem. Only the functions for LN and LP were changed; they were shifted slightly to force earlier movement into those sets. This change did not have much effect. While the azimuth limit error had the same value, the CMMCA actually offset earlier, more, and longer. It did turn back more about midway through the turn, and almost eliminated the heading divergence at end game.

5.2.4 Run #4, t_ahead = 60. Row #4 represents a logic change to stop the amount and length of the CMMCA's offset away from the CM. The change was performed exclusively to the t_now combining logic, and made the two components approximately equal in weight when combined. This was partially successful. The CMMCA did not offset excessively this time, although it was still slow to turn back towards the CM as the azimuth limit was exceeded. Also, the CMMCA did not offset until fairly late, thus it did not build turning room early. While this was due partly to the small look ahead value of only 60 seconds in this

simulation run, more likely the continuing problems were because of a errors introduced into the logic rules by mistake in this run.

5.2.5 Run #5, t_ahead = 60. The logic rule error was corrected. Because the offset and end game problems had been reduced both quantitatively and qualitatively, and because of the effect of the small look ahead time on the offset, the CMMCA was run through the much more complicated profile B of two linked 270 degrees turns (Row #5). Analysis of the results showed two problems. First, the CMMCA was not offsetting early enough (look ahead time, again) and then turning too far and too long. Second, and of major importance, the CMMCA was failing to offset properly following the first CM turn as the CM entered its second turn.

5.2.6 Run #6, t_ahead = 60. Row #6 shows the results following a change to all three factors. The range mapping was changed back to the original, where the scaling was fully reached at five and fifteen miles, not at the buffer boundaries. This effect would be better captured in the logic rules. The t_now combining logic was updated to do this.

The turn logic was reexamined and changed to end the excessive offset by the CMMCA that was causing the azimuth limit to be exceeded early in the profile. The t_now logic was modified slightly to increase the response to azimuths approaching the limits. The t_next fuzzy logic was changed significantly, in that no t_next outputs were allowed to be LN or LP. These changes forced the CMMCA to respond less to t_next outputs. Hence the CMMCA would still offset, but on a smaller scale, not turning away from the CM as much, and much more responsive to the need to turn back. These same changes allowed the CMMCA to create an offset for the next maneuver after completing the previous maneuver. Thus the CMMCA should now take better heed of CM maneuvers to set itself up prior to the CM going into its next maneuver.

Run #6 shows a large deviation in position and heading at the end of the run. However, it was again felt that this was due to end effects

as discussed for Run #2. If this is an end effect, then it should not cause problems in the transitions between CM maneuvers.

5.2.7 Run #7, t_ahead = 60, 90, 120. Because the results of the changes made in Run #6 were encouraging on the simple profile, profile C was tried. The results, as shown in Row #7, demonstrated a further ten percent improvement in azimuth results, even at the smallest look ahead value. As look ahead time increased, corresponding to allowing/forcing the CMMCA to act earlier, the CMMCA performance improved. A look ahead time of 120 seconds provided 100% coverage of the CM throughout the entire profile.

5.2.8 Run #8, t_ahead = 60, 90, 120. As a test, Profile C was rerun with the CM using 30 degrees of bank in all turns (Row #8). This resulted in a severe fall off in CMMCA performance at all values of look ahead time. The degradation was least severe at the smallest look ahead time of 60 seconds, probably due to the fact that the CM was turning so rapidly that the longer look ahead times were not allowing the CMMCA to respond quickly enough to CM maneuvers.

5.3 Computer Run Times

During early stages of the simulation, prior to incorporation of the fuzzy logic set point calculations the simulation ran much faster than real time on both the SPARCstation 2 and the PC. The later simulation efforts were carried out entirely on the PC. At the final form of the simulation, including all set point calculations, the system ran at almost exactly half of real time. That is, the shortest profile (profile A, 442 seconds of simulation time) ran in 15 minutes, while the longest (profile C, 1850 seconds of simulation time), took slightly over one hour. See Table 11 for more details.

The simulation run time appeared to be directly proportional to the length of the CM profile. This is a big improvement over the previous thesis efforts, where run time seemed to increase exponentially with profile length. The run time could be increased several fold by relatively minor changes. First, the Matlab code could be better

Table 11 CM Flight Durations and Simulation Times and Simulation Slow-Down Performance Ratio

Simulation Run Number	"Real" Time	Simulation Run Time	Ratio of Run Time to Real Time
1b	442.0	861.7	1.9:1
1c	442.0	939.2	2.1:1
2a	442.0	878.6	2.0:1
2b	442.0	860.4	1.9:1
2c	442.0	888.4	2.0:1
3a	442.0	855.6	1.9:1
4a	904.0	1774.9	2.0:1
5a	442.0	865.8	2.0:1
6a	1850.0	3500.9	1.9:1
7b	1850.0	3548.4	1.9:1
7c	1529.0	2964.2	1.9:1
8a	1529.0	2868.7	1.9:1
8b	1529.0	2948.7	1.9:1

optimized for speed. Second, converting the M-files to C or FORTRAN would speed up the simulation. Third, eliminating the unnecessary output (used for debugging, mostly) would also speed the simulation. Finally, using a workstation instead of a PC would probably speed up the simulation time by an order of magnitude.

An additional highly important factor affects this simulation as far as run time is concerned. Simulation overhead can take up a significant fraction of the actual CPU time used. To equate simulation run time, therefore, to the actual speed of an onboard aircraft autopilot would be seriously misleading. The onboard system would receive a real time data stream consisting of the CMMCA's inertial position, velocity, euler angles, radar range and gimbal angles, etc. The simulation's computation of these values would be eliminated, resulting in a huge reduction in the number of computations.

VI. Conclusions and Recommendations

6.1 Conclusions

This thesis has shown that the CMMCA tracking problem can be solved in nearly real time. With improvements to the computer code, as discussed in section 5.3, the simulation should be easily implementable in better than real time.

The new approaches to the CMMCA tracking problem taken in this thesis show good results and great promise. The keys to this approach were three-fold: 1) accept CM positions anywhere in the radar envelope instead of always forcing the CM back to the nominal desired position; 2) break the tracking into a series of short tracking solutions, changing every FTUT seconds; and 3) use fuzzy logic to compute the next set point.

Fuzzy logic succeeded because of a number of factors. It allowed an accurate identification of the problem and used workable input-output relations (i.e., HCA, azimuth, etc.). The analog to scale mappings and the set membership functions were also workable and proper. Interrelationships between all variables remained consistent throughout the fuzzy logic development. Finally, the logic rule tables were kept small and simple, ensuring that they remained clear, intuitive, and easy to modify.

This is not to say that no improvements are possible. Different or additional variables should be considered, along with different values of FTUT. The shape and position of the fuzzy set membership functions and the mappings from analog to scaled values are also likely candidates for change and improvement. All these factors had a significant impact on the CMMCA flight path, and consequently on the CM radar coverage provided.

The results in the last two rows of Table 10 show the changes that occur as the look ahead time is varied. It is not clear what value is

best; an experienced pilot sub-consciously uses many look ahead times, up to several minutes in the future. Therefore, it seems reasonable to believe that instead of one optimal look ahead time, significant improvement would occur if multiple look ahead times were incorporated, perhaps based on CM maneuvers, but more appropriately within the fuzzy logic decision tables themselves.

There are at least two consequences of the fact that the *true* parameter values were not used: 1) the true values will need to be found and input for actual implementation in autopilot form; 2) the results may be biased in some unknown direction, or may differ in a random fashion from truth. Hence the simulation will need to be closely scrutinized following the incorporation of these true parameter values.

The penalty function and dynamic programming approaches to this problem suffer from major difficulties that have been eliminated here. The solution time varies directly with profile length, not exponentially. This is because the former approaches used iterative methods seeking solution convergence. This method does not iterate, so does not have problems caused by failure of the solutions to converge.

6.2 Recommendations

6.2.1 Program Input. The input method consisted of an ASCII file of circular and straight segments. The C_m was assumed to be in constant bank while turning and to have constant airspeed. A user friendly, efficient flight path data input method is needed, based on the actual information provided to the aircrews. Emphasis must be placed on making both the data entry and simulation results simple, quick and intuitive; if not, then aircrews will not use this program.

6.2.2 Fuzzy Logic. The fuzzy logic developed in this thesis works well for CM profiles at small bank angles. However, the solution breaks down as the CM bank increases. As discussed above, the fuzzy logic needs to be expanded to several look ahead times. A reasonable approach might be to use 60 and 90 seconds or 60, 90, and 120 seconds; further experimentation should provide guidance.

A second means of improving the fuzzy logic would be to change the mapping from the analog to the scaled [-5,5] values. Also, changing the locations of the fuzzy set membership functions, or perhaps adding two more functions for Medium Positive (MP) and Medium Negative (MN) might help.

6.2.3 CMMCA Performance. The CMMCA had no problem tracking the CM when the CM maintained a constant airspeed. However, the fuzzy logic should be modified to allow the CMMCA to change speed, but still attempt to keep speed changes small. Since turn radius depends on the square of the speed, relatively small changes in speed might greatly increase the CMMCA's ability to track the CM.

One other problem occurred with the CMMCA turn performance. The CMMCA bank angle accelerations were occasionally very oscillatory, usually when the set point commanded bank angle was near zero. As soon as the CMMCA turned slightly, the command would reverse direction (see the discussion in section 2.6.4). This could be overcome by either a second set of fuzzy logic tables, or by limiting the CMMCA's maximum bank angle or bank angle acceleration when near the desired state.

This thesis used a model with no noise present. Adding noise, especially to the range and angles supplied by the radar, would greatly increase the realism of the simulation.

6.2.4 Simulation Times. The values of sample time (1 second), FTUT (10 seconds) and look ahead time (60, 90, or 120 seconds) were chosen for convenience and because they seemed about the right size. There is no reason to believe that other values might not work better; additional effort should go into an examination of different simulation times.

Appendix A: Plots of Simulation Output

Each figure is a plot of the CM and CMMCA ground track over the entire simulation run time. In addition, a line is drawn at intervals of 60 seconds from the CMMCA to the CM. This aids in interpreting the plot, and also graphically demonstrates the range and azimuth at one minute intervals.

The symbology on each graph is the same:

CMMCA ground track: solid line;
"o" marks the CMMCA position every 60 seconds

CM ground track: dot-dashed line;
"x" marks the CM position every 60 seconds

Line of sight from CMMCA to CM: dotted line;
shown every 60 seconds

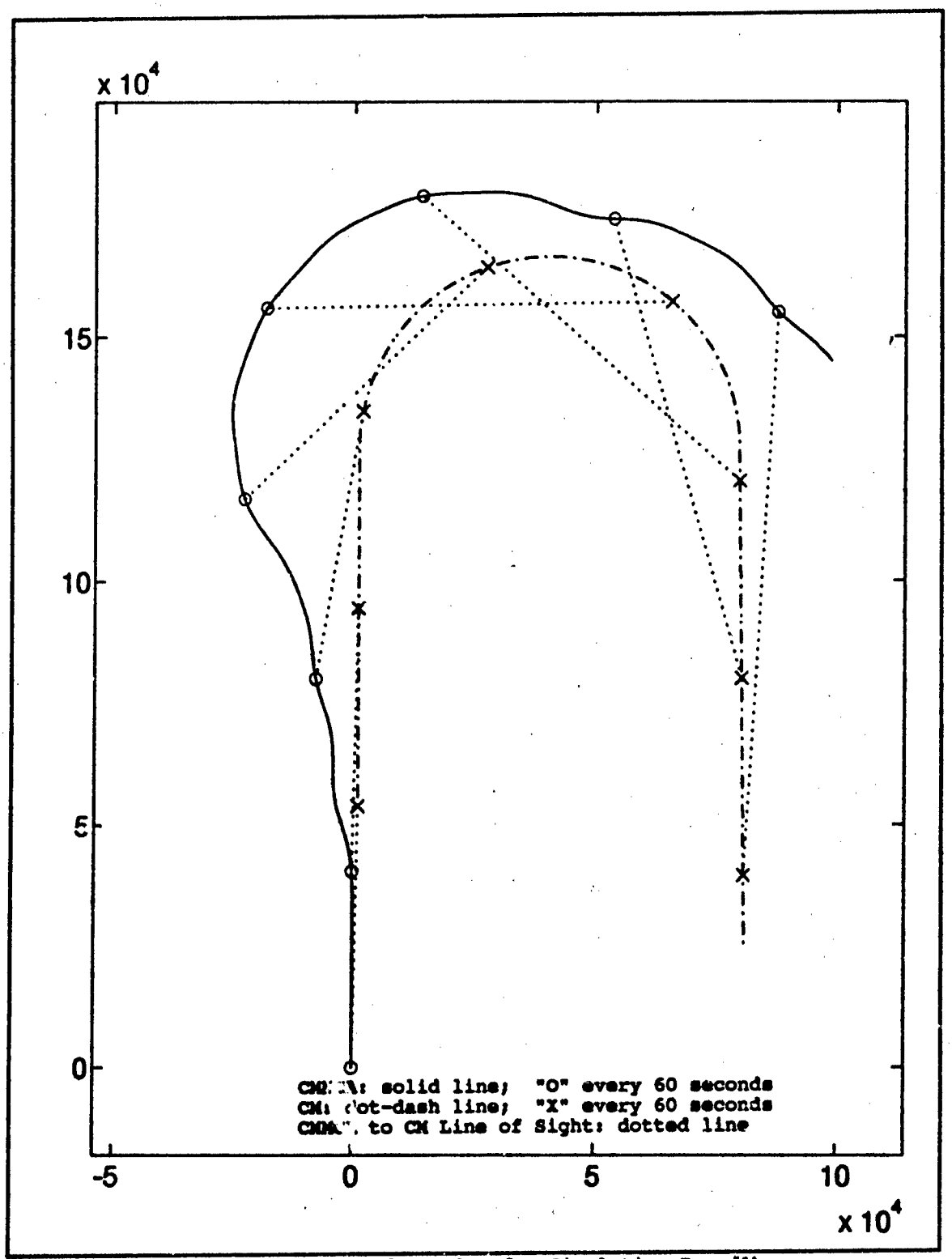

Figure 23 Plot of Results for Simulation Run #1b
Look Ahead Time = 90 seconds

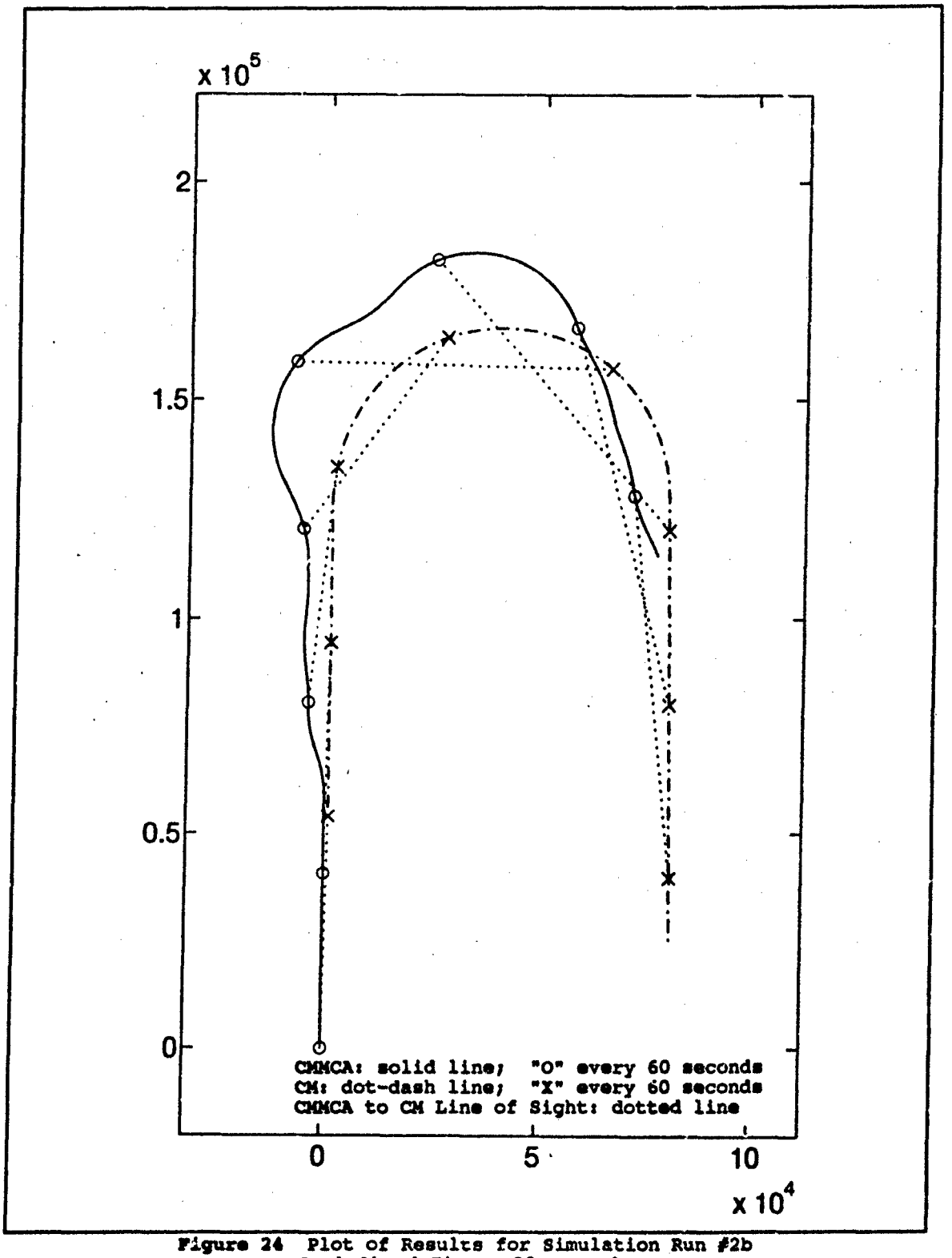

Figure 24 Plot of Results for Simulation Run #2b
Look Ahead Time = 90 seconds

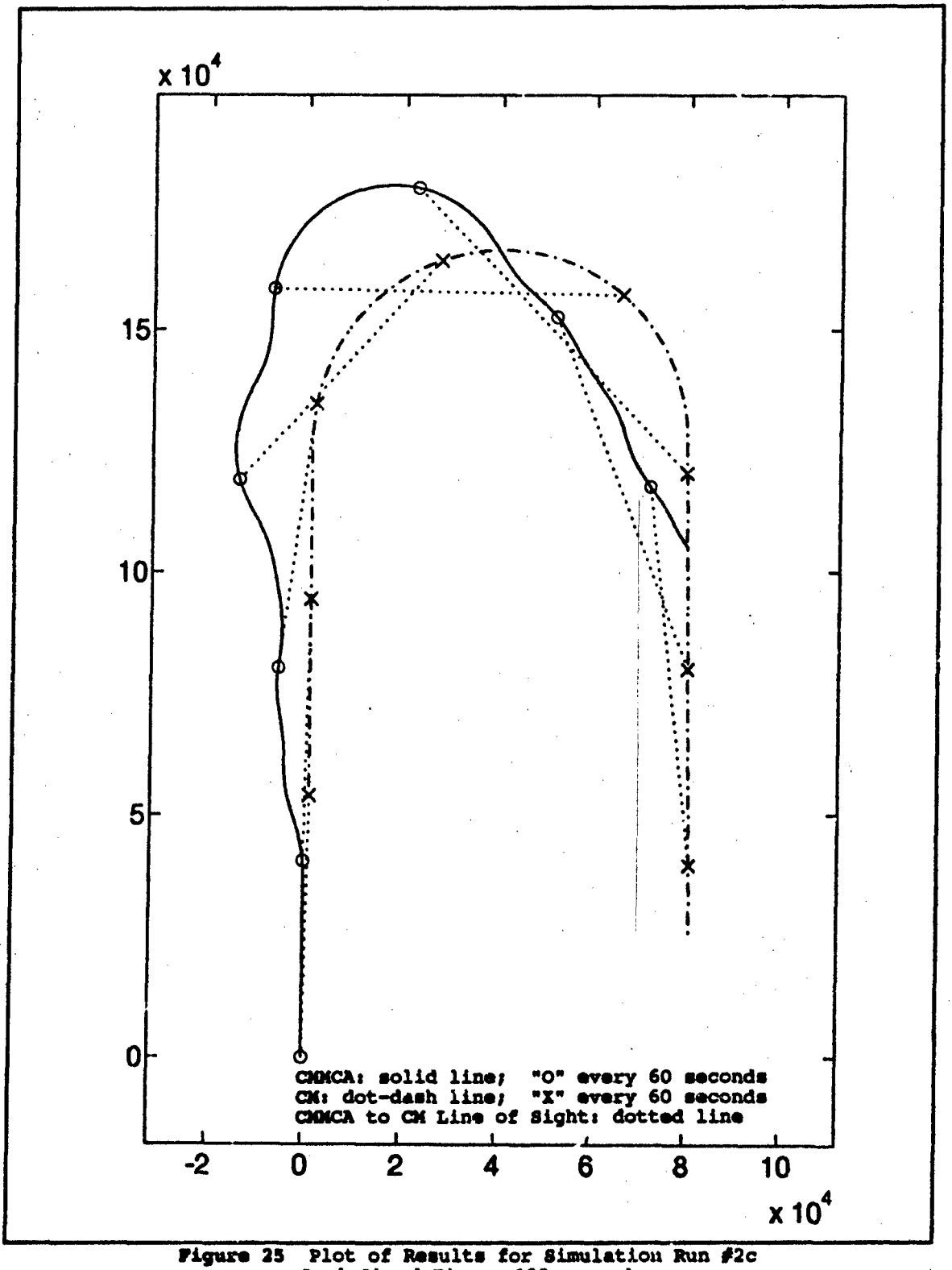

Figure 25 Plot of Results for Simulation Run #2c
Look Ahead Time = 120 seconds

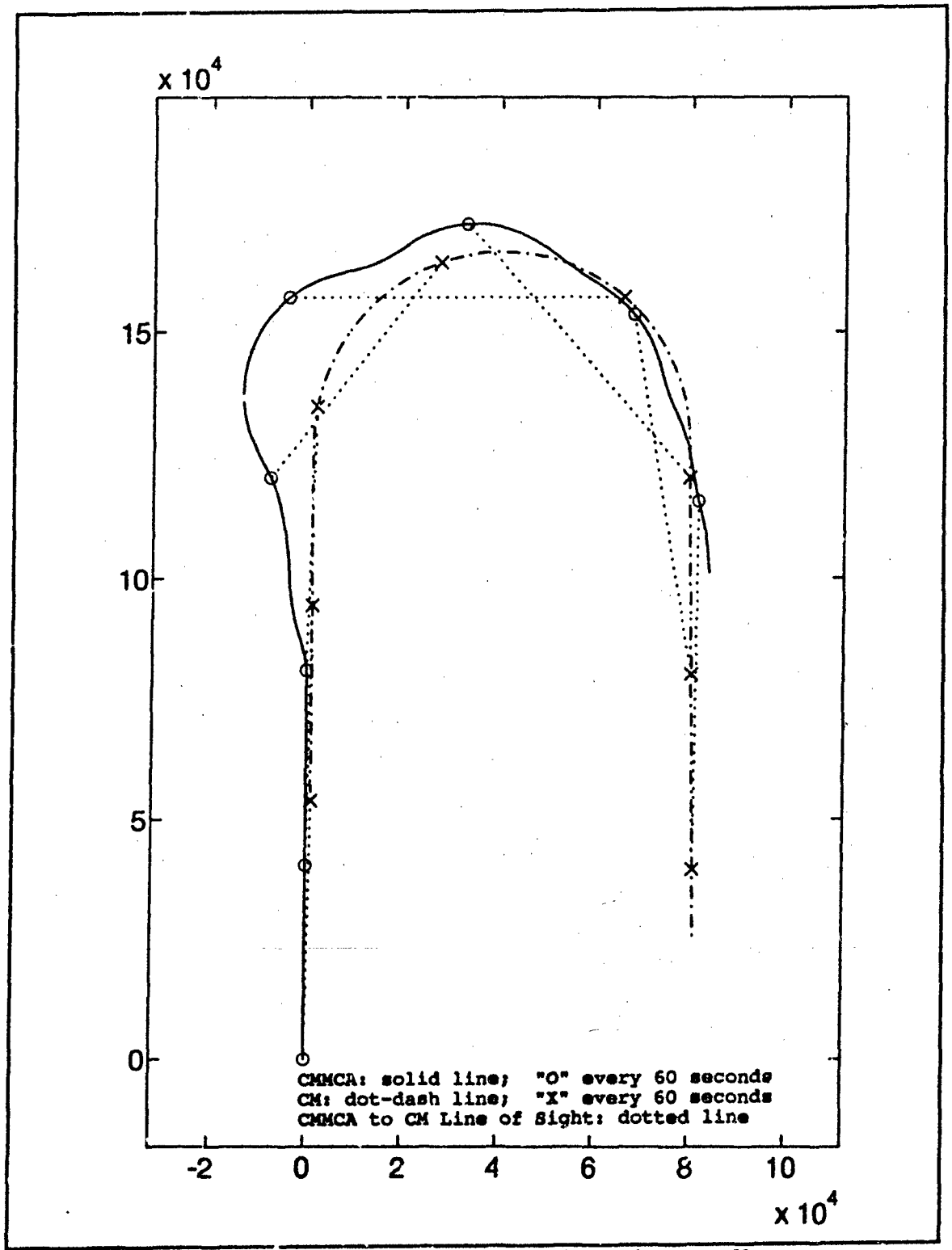

Figure 26 Plot of Results for Simulation Run #3a
Look Ahead Time = 60 seconds

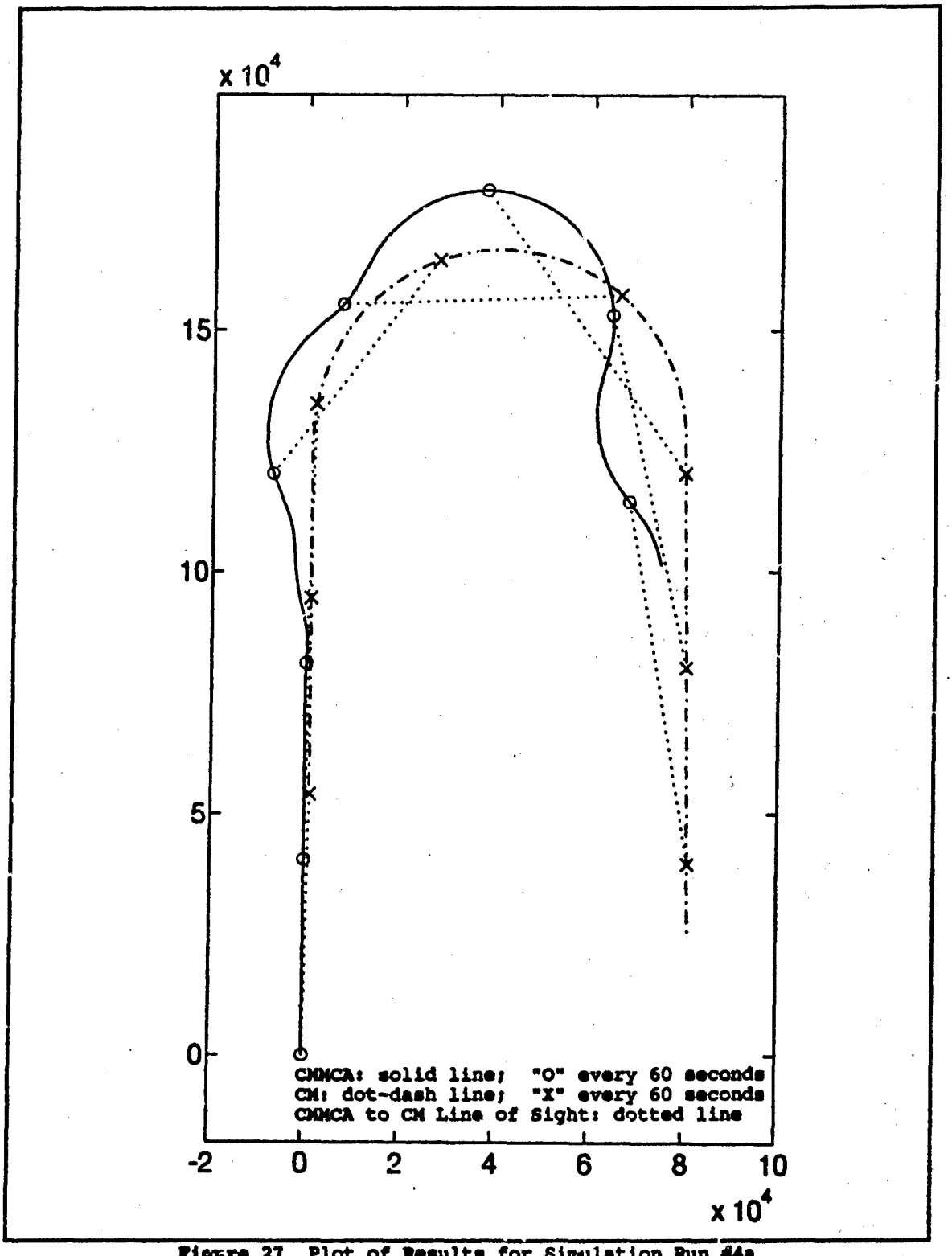

Figure 27 Plot of Results for Simulation Run #4a
Look Ahead Time = 60 seconds

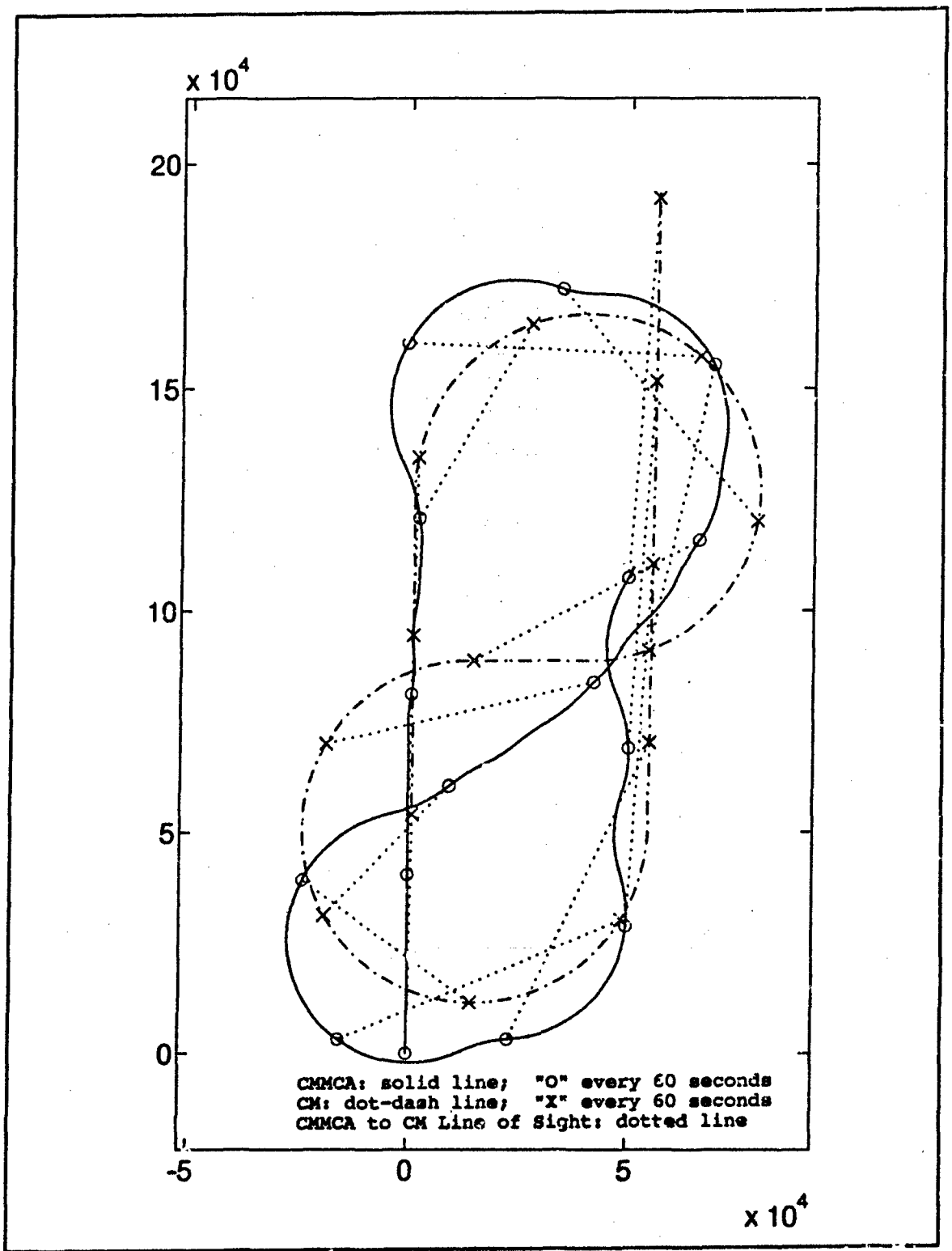

Figure 28 Plot of Results for Simulation Run #5a
Look Ahead Time = 60 seconds

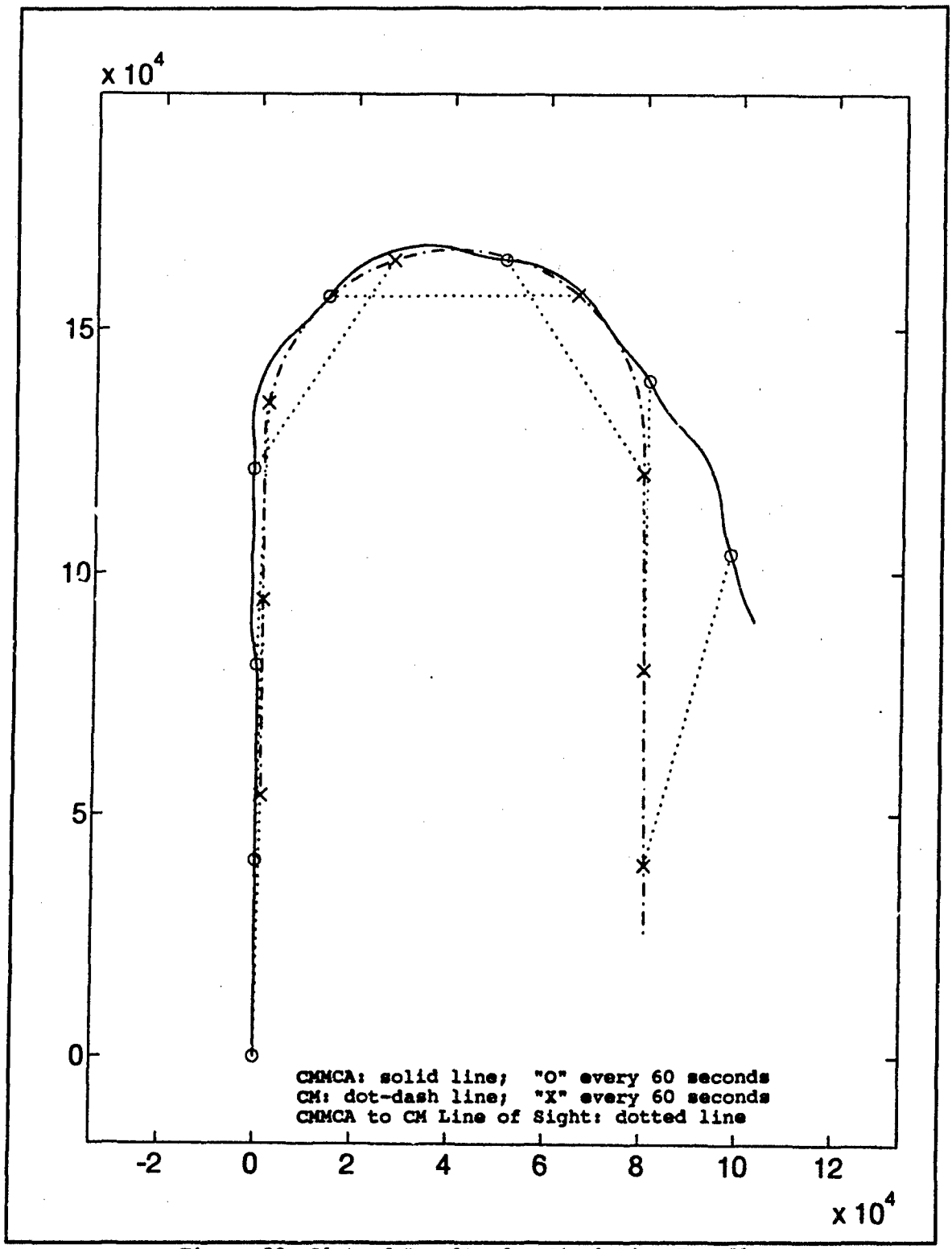

Figure 29 Plot of Results for Simulation Run #6a
Look Ahead Time = 60 seconds

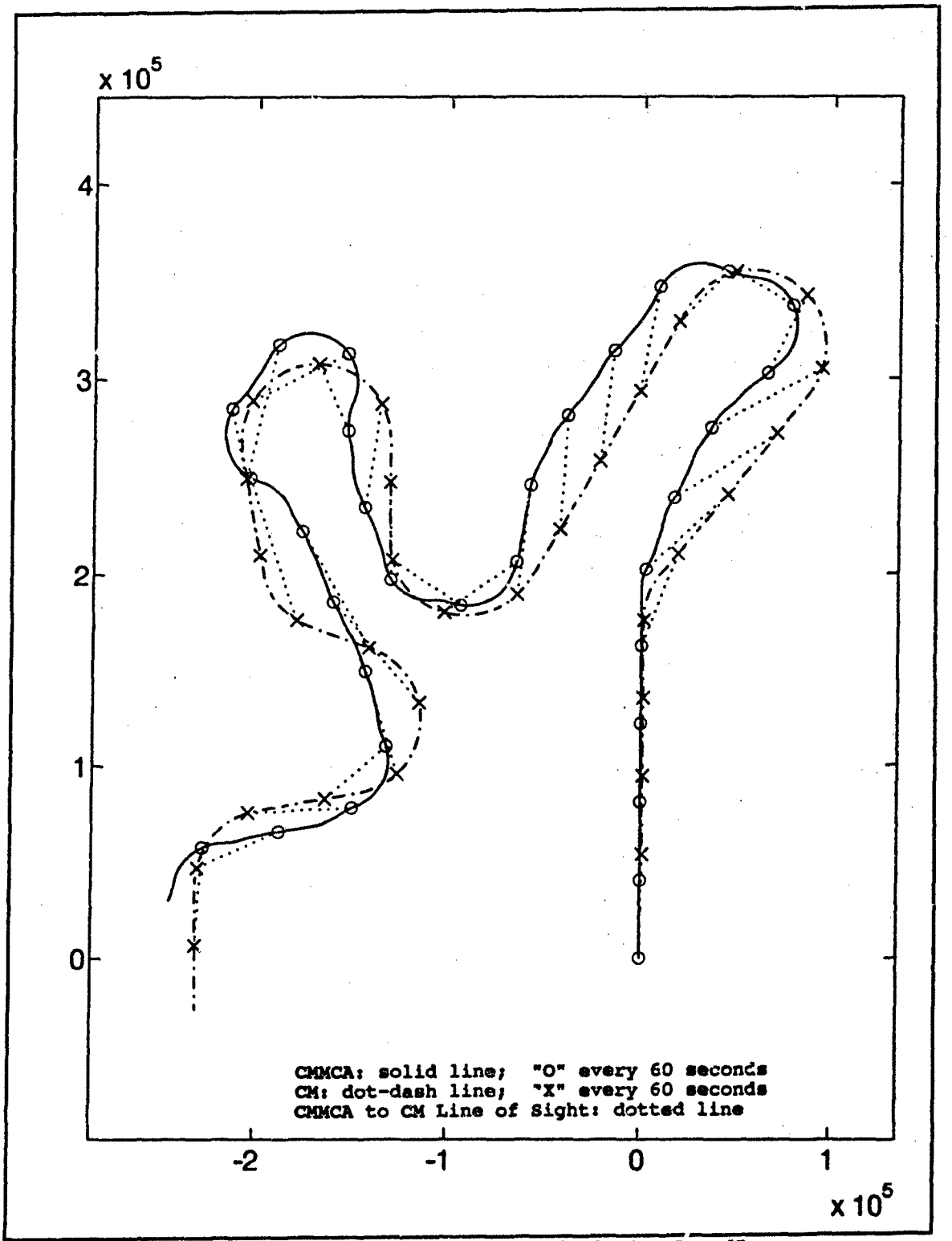

Figure 30 Plot of Results for Simulation Run #7a
Look Ahead Time = 60 seconds

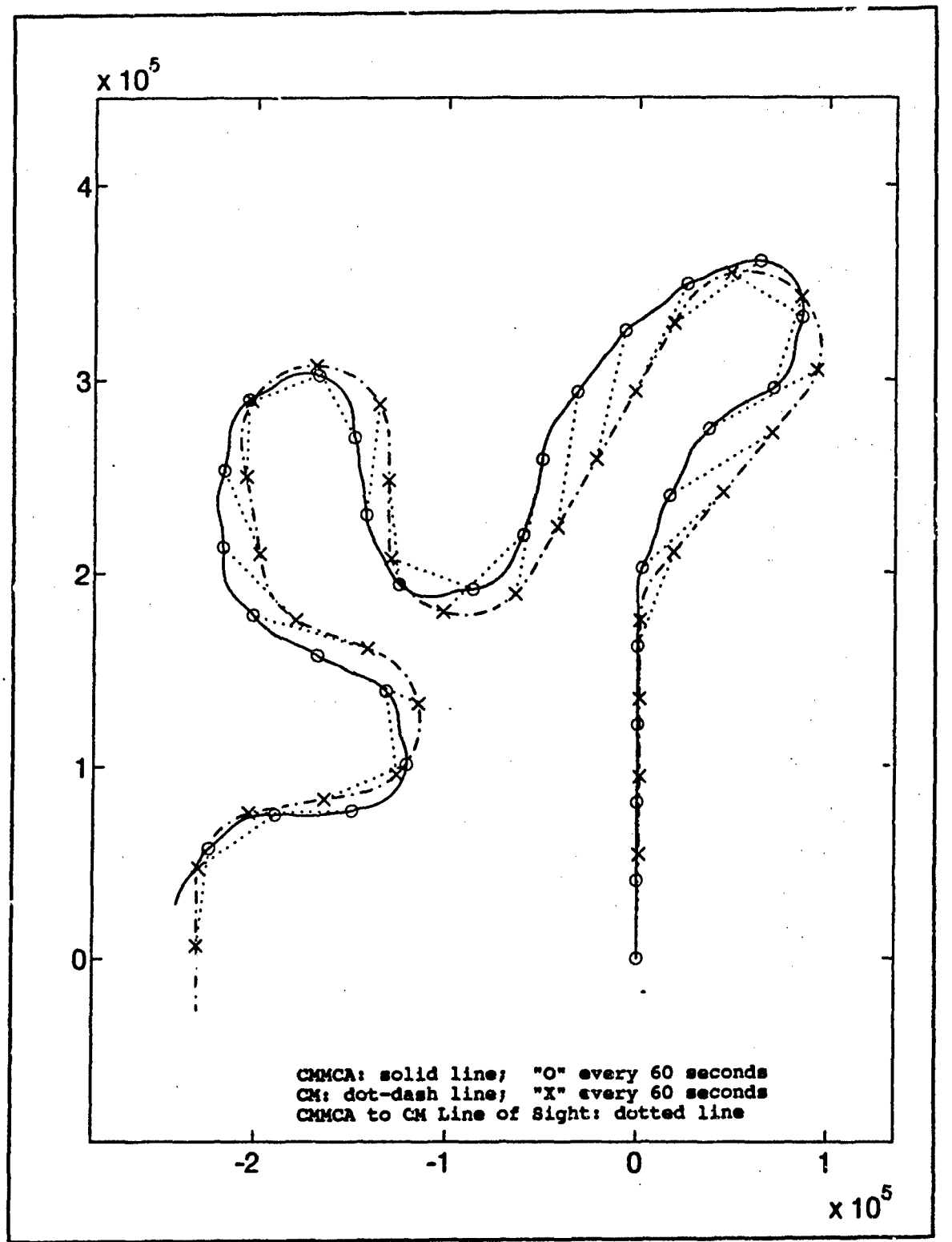

Figure 31 Plot of Results for Simulation Run #7b
Look Ahead Time = 90 seconds

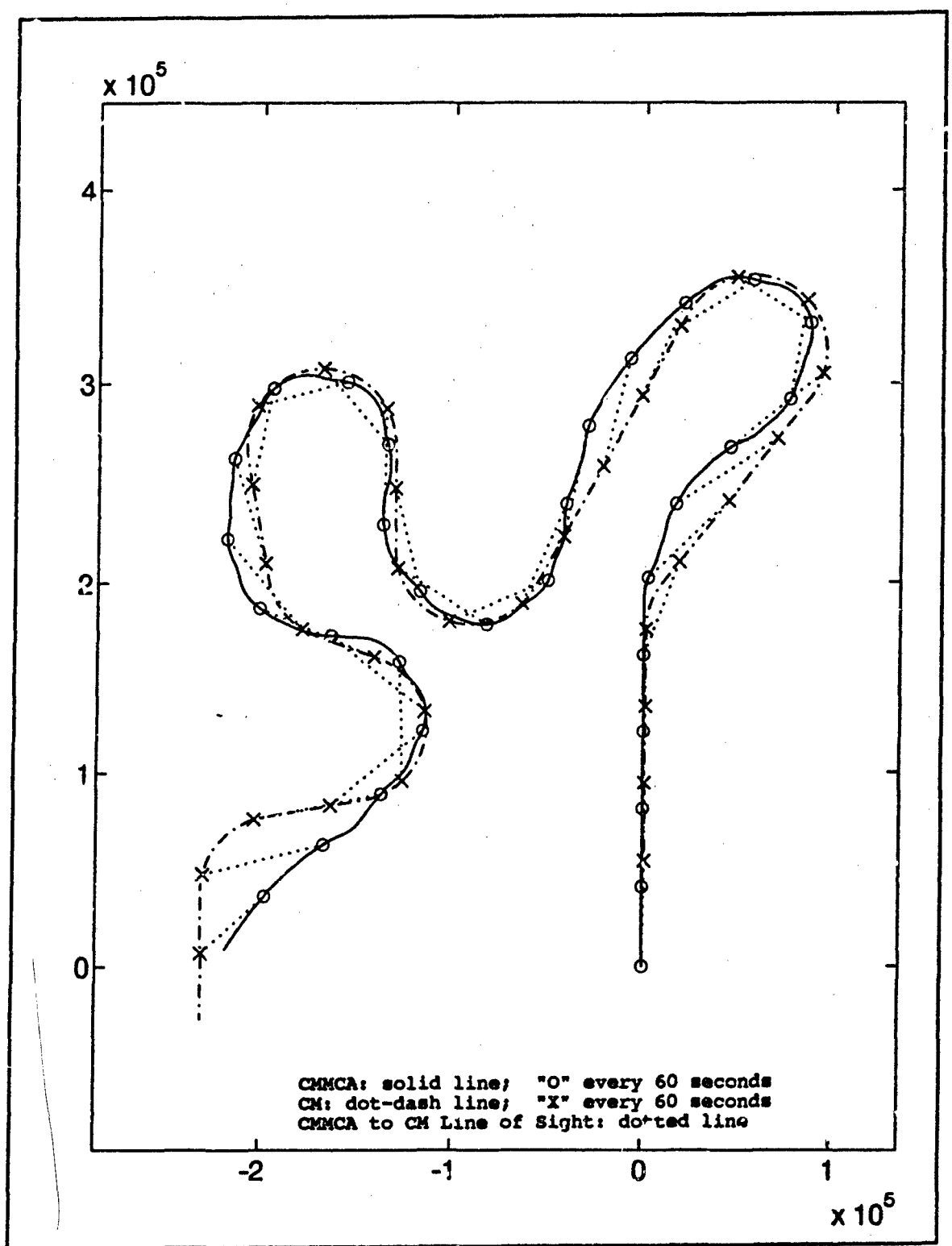

Figure 32 Plot of Results for Simulation Run #7c
Look Ahead Time = 120 seconds

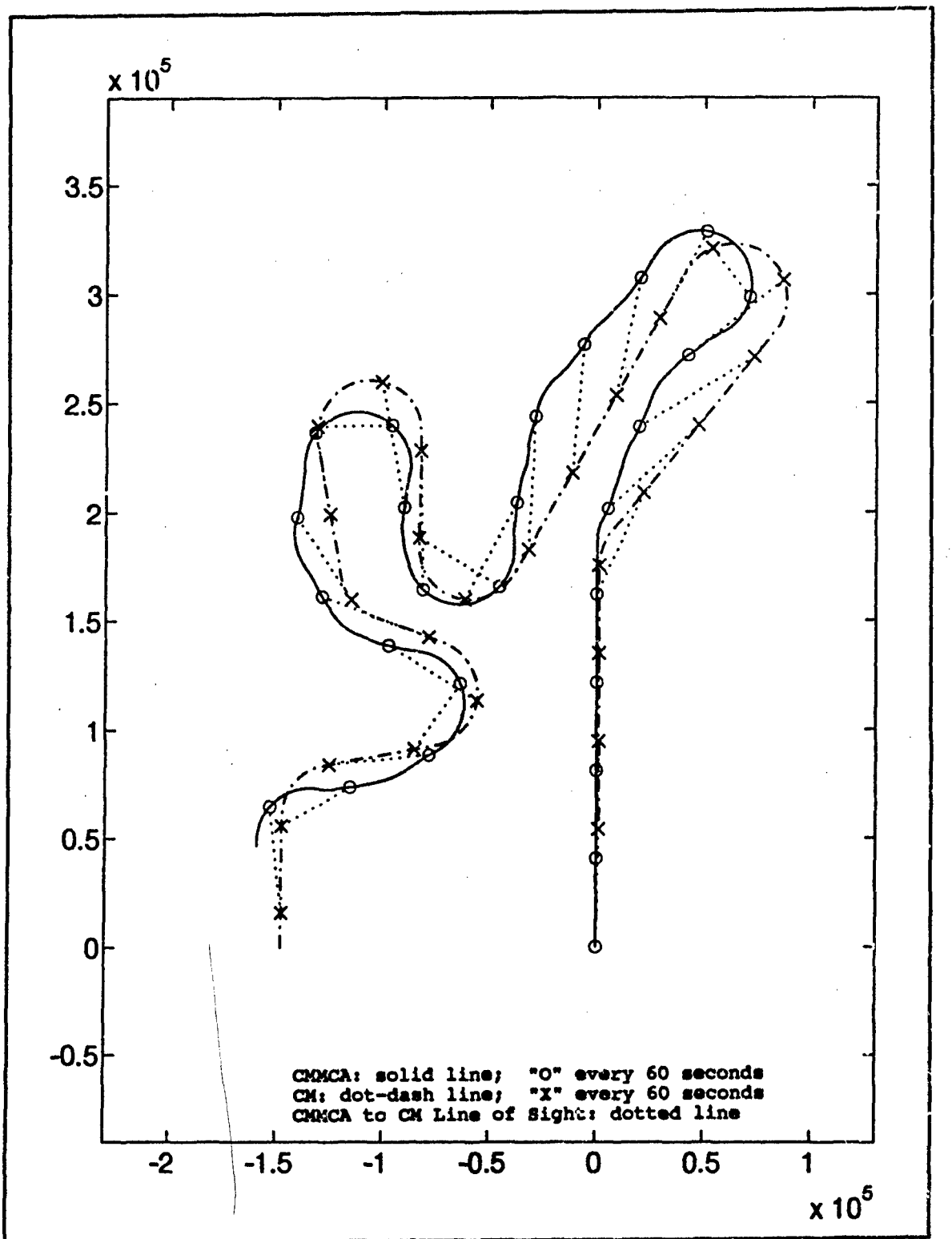

Figure 33 Plot of Results for Simulation Run #8a
Look Ahead Time = 60 seconds

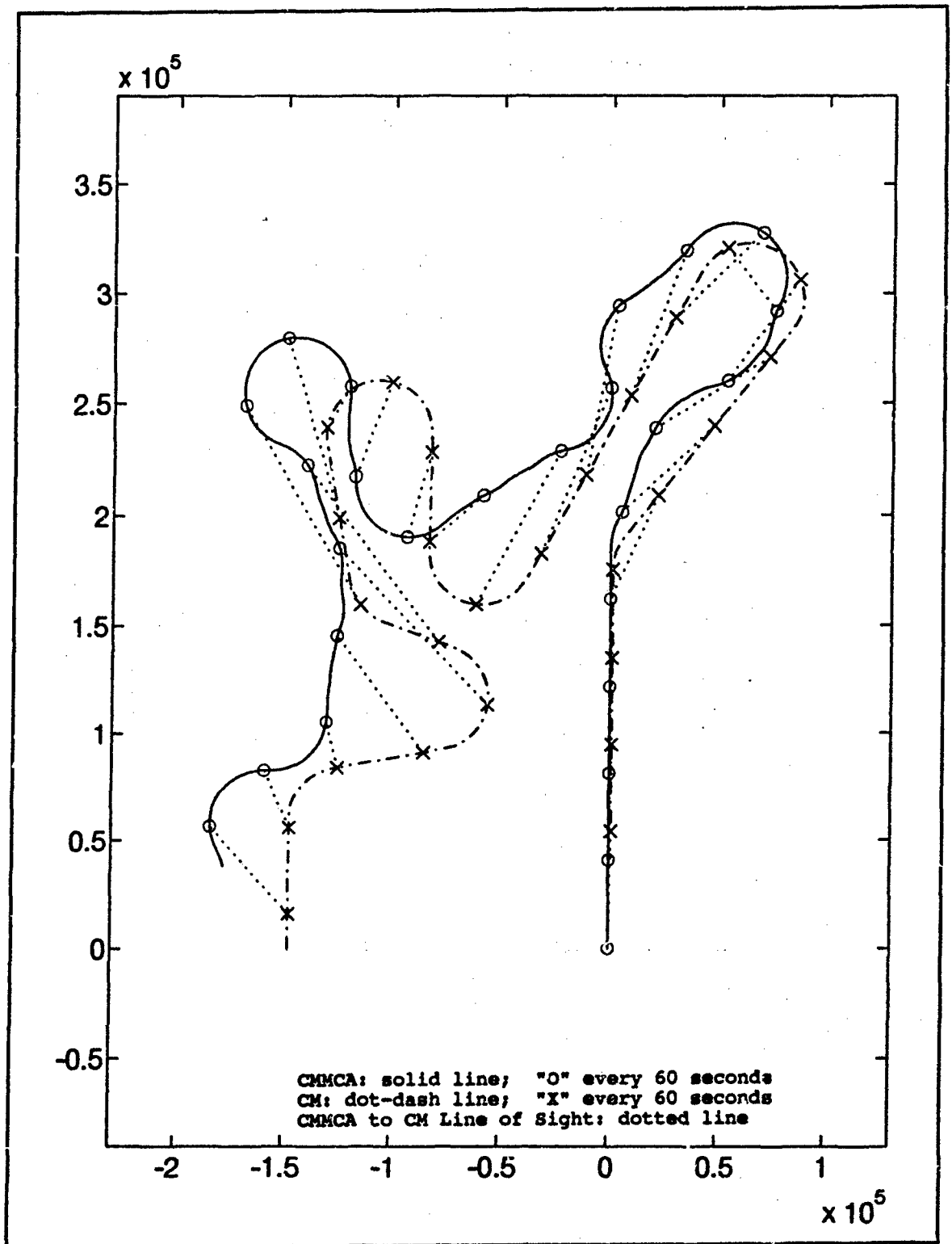

Figure 34 Plot of Results for Simulation Run #8b
Look Ahead Time = 90 seconds

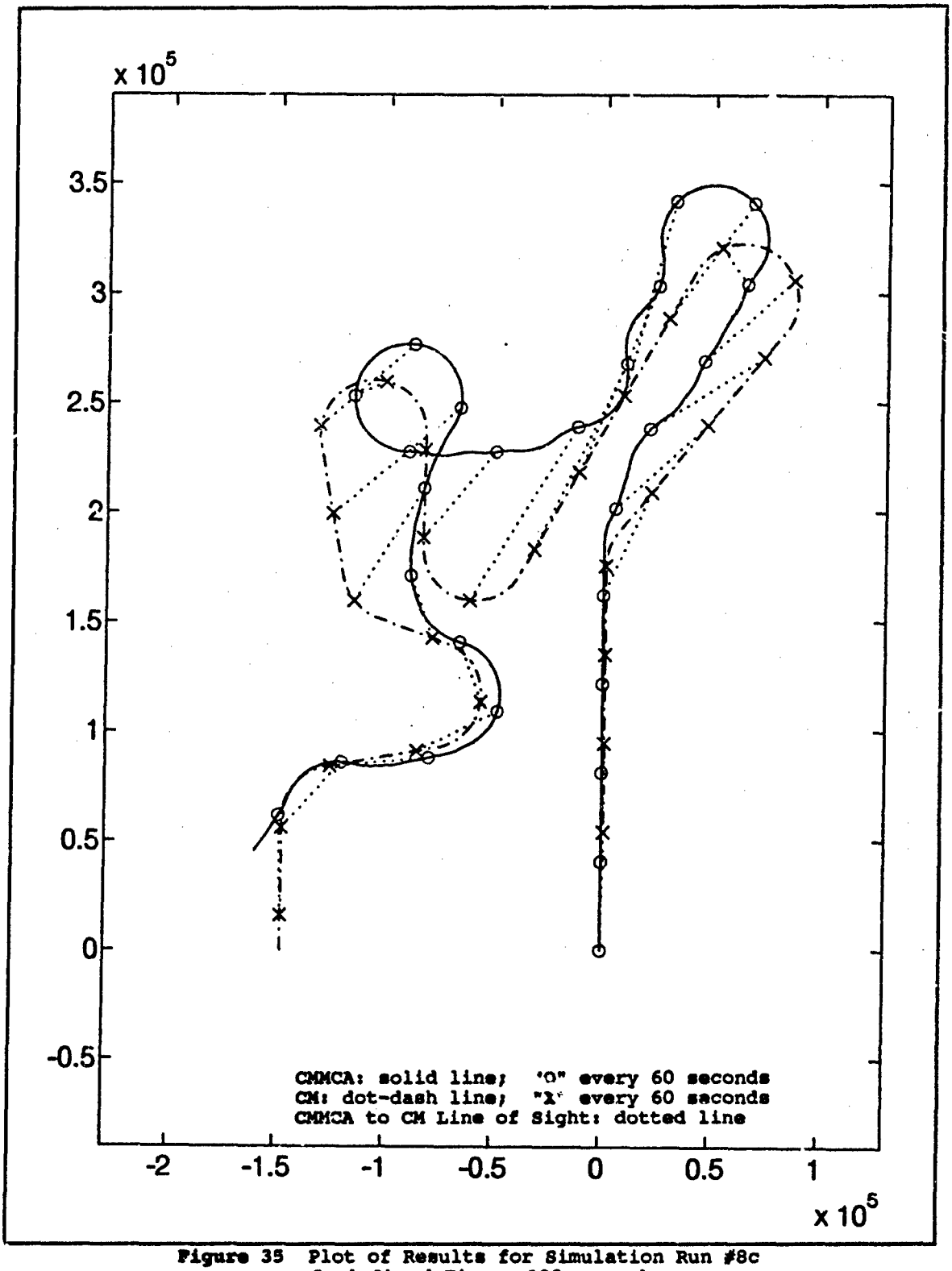

Figure 35 Plot of Results for Simulation Run #8c
Look Ahead Time = 120 seconds

Appendix B: Numerical Output of the Simulation

The full output from one simulation run of profile C is 326K bytes. Because the amount of data is immense, only a small, representative sample of the output for Run #7c is presented here.

Sample of the CM states:

Time	Position			Bank	Velocity			Roll
	X	Y	Z	Angle	X	Y	Z	Rate
1234	267911.17	-208467.09	-1000.00	0.00	-676.15	12.66	0.00	0.00
1235	267235.15	-208448.58	-1000.00	0.00	-675.85	24.36	0.00	0.00
1236	266559.53	-208418.37	-1000.00	0.00	-675.35	36.06	0.00	0.00
1237	265884.51	-208376.46	-1000.00	0.00	-674.65	47.76	0.00	0.00
1238	265210.29	-208322.85	-1000.00	0.00	-673.75	59.47	0.00	0.00
1239	264537.08	-208257.52	-1000.00	0.00	-672.64	71.18	0.00	0.00
1240	263865.07	-208180.49	-1000.00	0.00	-671.33	82.89	0.00	0.00
1241	263194.48	-208091.74	-1000.00	0.00	-669.82	94.61	0.00	0.00
1242	262525.50	-207991.27	-1000.00	0.00	-668.10	106.33	0.00	0.00
1243	261858.34	-207879.07	-1000.00	0.00	-666.18	118.06	0.00	0.00
1244	261192.84	-207757.10	-1000.00	0.00	-665.17	123.93	0.00	0.00
1245	260527.67	-207633.17	-1000.00	0.00	-665.17	123.93	0.00	0.00
1246	259862.50	-207509.25	-1000.00	0.00	-665.17	123.93	0.00	0.00
1247	259197.33	-207385.32	-1000.00	0.00	-665.17	123.93	0.00	0.00
1248	258532.16	-207261.40	-1000.00	0.00	-665.17	123.93	0.00	0.00
1249	257866.99	-207137.47	-1000.00	0.00	-665.17	123.93	0.00	0.00
1250	257201.82	-207013.55	-1000.00	0.00	-665.17	123.93	0.00	0.00
1251	256536.65	-206889.62	-1000.00	0.00	-665.17	123.93	0.00	0.00
1252	255871.48	-206765.70	-1000.00	0.00	-665.17	123.93	0.00	0.00
1253	255206.31	-206641.77	-1000.00	0.00	-665.17	123.93	0.00	0.00
1254	254541.14	-206517.85	-1000.00	0.00	-665.17	123.93	0.00	0.00
1255	253875.97	-206393.92	-1000.00	0.00	-665.17	123.93	0.00	0.00
1256	253210.80	-206270.00	-1000.00	0.00	-665.17	123.93	0.00	0.00
1257	252545.63	-206146.07	-1000.00	0.00	-665.17	123.93	0.00	0.00
1258	251880.46	-206022.15	-1000.00	0.00	-665.17	123.93	0.00	0.00
1259	251215.29	-205898.22	-1000.00	0.00	-665.17	123.93	0.00	0.00
1260	250550.12	-205774.30	-1000.00	0.00	-665.17	123.93	0.00	0.00
1261	249884.95	-205650.37	-1000.00	0.00	-665.17	123.93	0.00	0.00
1262	249219.78	-205526.45	-1000.00	0.00	-665.17	123.93	0.00	0.00
1263	248554.61	-205402.52	-1000.00	0.00	-665.17	123.93	0.00	0.00
1264	247889.44	-205278.60	-1000.00	0.00	-665.17	123.93	0.00	0.00
1265	247224.27	-205154.67	-1000.00	0.00	-665.17	123.93	0.00	0.00
1266	246559.10	-205030.75	-1000.00	0.00	-665.17	123.93	0.00	0.00
1267	245893.93	-204906.82	-1000.00	0.00	-665.17	123.93	0.00	0.00
1268	245228.76	-204782.90	-1000.00	0.00	-665.17	123.93	0.00	0.00
1269	244563.59	-204658.97	-1000.00	0.00	-665.17	123.93	0.00	0.00
1270	243898.42	-204535.05	-1000.00	0.00	-665.17	123.93	0.00	0.00
1271	243233.25	-204411.12	-1000.00	0.00	-665.17	123.93	0.00	0.00
1272	242568.08	-204287.20	-1000.00	0.00	-665.17	123.93	0.00	0.00
1273	241902.91	-204163.27	-1000.00	0.00	-665.17	123.93	0.00	0.00
1274	241237.74	-204039.35	-1000.00	0.00	-665.17	123.93	0.00	0.00
1275	240572.57	-203915.42	-1000.00	0.00	-665.17	123.93	0.00	0.00
1276	239907.40	-203791.50	-1000.00	0.00	-665.17	123.93	0.00	0.00
1277	239242.23	-203667.57	-1000.00	0.00	-665.17	123.93	0.00	0.00
1278	238577.06	-203543.64	-1000.00	0.00	-665.17	123.93	0.00	0.00

Sample of the CMMCA states:

Time	Position			Bank	Velocity			Roll
	X	Y	Z	Angle	X	Y	Z	Rate
1234	304282.98	-178146.17	-29000.00	-30.13	-14.81	-674.94	0.00	0.10
1235	304258.84	-178820.82	-29000.00	-30.05	-33.45	-674.27	0.00	0.06
1236	304216.10	-179494.54	-29000.00	-30.00	-52.02	-673.09	0.00	0.03
1237	304154.82	-180166.83	-29000.00	-29.98	-70.53	-671.40	0.00	0.01
1238	304075.06	-180837.18	-29000.00	-29.98	-88.97	-669.21	0.00	0.00
1239	303976.89	-181505.08	-29000.00	-29.98	-107.35	-666.51	0.00	0.00
1240	303860.38	-182170.03	-29000.00	-29.98	-125.65	-663.30	0.00	-0.01
1241	303725.66	-182831.53	-29000.00	-29.40	-143.73	-659.62	0.00	1.40
1242	303573.21	-183489.17	-29000.00	-28.56	-161.04	-655.61	0.00	0.37
1243	303403.70	-184142.62	-29000.00	-28.49	-177.95	-651.22	0.00	-0.15
1244	303217.32	-184791.46	-29000.00	-28.76	-194.83	-646.37	0.00	-0.35
1245	303014.00	-185435.20	-29000.00	-29.13	-211.81	-641.01	0.00	-0.37
1246	302793.65	-186073.30	-29000.00	-29.47	-228.90	-635.11	0.00	-0.30
1247	302556.19	-186705.24	-29000.00	-29.73	-246.03	-628.67	0.00	-0.21
1248	302301.61	-187330.47	-29000.00	-29.89	-263.12	-621.71	0.00	-0.13
1249	302029.99	-187948.50	-29000.00	-29.99	-280.10	-614.25	0.00	-0.07
1250	301741.47	-188558.81	-29000.00	-30.03	-296.91	-606.30	0.00	-0.03
1251	301436.23	-189160.94	-29000.00	-30.04	-313.52	-597.88	0.00	0.00
1252	301114.50	-189754.42	-29000.00	-30.04	-329.89	-589.01	0.00	0.01
1253	300776.53	-190338.80	-29000.00	-30.03	-346.01	-579.69	0.00	0.01
1254	300422.57	-190913.65	-29000.00	-30.02	-361.86	-569.93	0.00	0.01
1255	300052.91	-191478.52	-29000.00	-30.01	-377.42	-559.74	0.00	0.01
1256	299667.83	-192032.99	-29000.00	-30.01	-392.70	-549.13	0.00	0.00
1257	299267.62	-192576.65	-29000.00	-30.00	-407.67	-538.11	0.00	0.00
1258	298852.59	-193109.08	-29000.00	-30.00	-422.33	-526.68	0.00	0.00
1259	298423.06	-193629.88	-29000.00	-30.00	-436.67	-514.85	0.00	0.00
1260	297979.36	-194138.66	-29000.00	-30.00	-450.68	-502.64	0.00	0.00
1261	297521.81	-194635.03	-29000.00	-29.92	-464.35	-490.04	0.00	0.74
1262	297050.85	-195118.68	-29000.00	-29.45	-477.48	-477.26	0.00	0.24
1263	296567.02	-195589.47	-29000.00	-29.35	-490.12	-464.27	0.00	-0.02
1264	296070.73	-196047.10	-29000.00	-29.45	-502.41	-450.93	0.00	-0.14
1265	295562.30	-196491.20	-29000.00	-29.60	-514.41	-437.20	0.00	-0.16
1266	295042.02	-196921.37	-29000.00	-29.75	-526.10	-423.06	0.00	-0.13
1267	294510.21	-197337.20	-29000.00	-29.86	-537.46	-408.53	0.00	-0.10
1268	293967.22	-197738.31	-29000.00	-29.94	-548.46	-393.64	0.00	-0.06
1269	293413.42	-198124.36	-29000.00	-29.99	-559.08	-378.41	0.00	-0.03
1270	292849.20	-198495.02	-29000.00	-30.01	-569.28	-362.87	0.00	-0.01
1271	292274.99	-198850.00	-29000.00	-30.02	-579.07	-347.05	0.00	0.00
1272	291691.24	-199189.05	-29000.00	-29.44	-588.35	-331.07	0.00	1.29
1273	291098.60	-199512.34	-29000.00	-27.52	-596.75	-315.67	0.00	2.51
1274	290498.14	-199820.85	-29000.00	-24.50	-603.96	-301.64	0.00	3.50
1275	289891.12	-200116.28	-29000.00	-20.57	-609.83	-289.59	0.00	4.32
1276	289278.93	-200400.83	-29000.00	-15.91	-614.33	-279.93	0.00	4.99
1277	288662.93	-200677.04	-29000.00	-10.64	-617.45	-272.97	0.00	5.53
1278	288044.48	-200947.74	-29000.00	-4.88	-619.22	-268.93	0.00	5.93
1279	287424.94	-201215.93	-29000.00	1.29	-619.63	-267.99	0.00	6.35
1280	286805.69	-201484.80	-29000.00	7.79	-618.61	-270.32	0.00	6.65

Sample of the CMMCA euler angles:

Time	Heading	Pitch	Bank
1234	268.74	0.00	-30.13
1235	267.16	0.00	-30.05
1236	265.59	0.00	-30.00
1237	264.00	0.00	-29.98
1238	262.43	0.00	-29.98
1239	260.85	0.00	-29.98
1240	259.27	0.00	-29.98
1241	257.71	0.00	-29.40
1242	256.20	0.00	-28.56
1243	254.72	0.00	-28.49
1244	253.23	0.00	-28.76
1245	251.71	0.00	-29.13
1246	250.18	0.00	-29.47
1247	248.63	0.00	-29.73
1248	247.06	0.00	-29.89
1249	245.49	0.00	-29.99
1250	243.91	0.00	-30.03
1251	242.33	0.00	-30.04
1252	240.75	0.00	-30.04
1253	239.17	0.00	-30.03
1254	237.59	0.00	-30.02
1255	236.01	0.00	-30.01
1256	234.43	0.00	-30.01
1257	232.85	0.00	-30.00
1258	231.27	0.00	-30.00
1259	229.70	0.00	-30.00
1260	228.12	0.00	-30.00
1261	226.54	0.00	-29.92
1262	224.99	0.00	-29.45
1263	223.45	0.00	-29.35
1264	221.91	0.00	-29.45
1265	220.36	0.00	-29.60
1266	218.80	0.00	-29.75
1267	217.24	0.00	-29.85
1268	215.67	0.00	-29.94
1269	214.09	0.00	-29.99
1270	212.51	0.00	-30.01
1271	210.94	0.00	-30.02
1272	209.37	0.00	-29.44
1273	207.88	0.00	-27.52
1274	206.54	0.00	-24.50
1275	205.40	0.00	-20.57
1276	204.50	0.00	-15.91
1277	203.85	0.00	-10.64
1278	203.48	0.00	-4.88
1279	203.39	0.00	1.29
1280	203.60	0.00	7.79

Sample of the CMMCA to CM range and radar gimbal angles:

Time	Range	Azimuth	Elevation
1234	55011.51	-55.30	-6.57
1235	55068.66	-54.93	-6.73
1236	55123.54	-54.55	-6.86
1237	55176.08	-54.18	-6.98
1238	55226.19	-53.80	-7.09
1239	55273.79	-53.42	-7.19
1240	55318.79	-53.03	-7.30
1241	55361.13	-52.60	-7.88
1242	55400.97	-52.18	-8.65
1243	55438.58	-51.84	-8.80
1244	55475.02	-51.53	-8.68
1245	55514.31	-51.19	-8.50
1246	55557.03	-50.82	-8.36
1247	55602.67	-50.42	-8.30
1248	55650.76	-49.97	-8.32
1249	55700.88	-49.50	-8.41
1250	55752.65	-49.01	-8.54
1251	55805.72	-48.50	-8.71
1252	55859.78	-47.97	-8.90
1253	55914.50	-47.42	-9.10
1254	55969.61	-46.86	-9.30
1255	56024.81	-46.29	-9.52
1256	56079.83	-45.70	-9.74
1257	56134.42	-45.09	-9.98
1258	56188.33	-44.47	-10.22
1259	56241.32	-43.84	-10.47
1260	56293.17	-43.19	-10.73
1261	56343.66	-42.51	-11.05
1262	56392.67	-41.78	-11.64
1263	56440.12	-41.10	-11.98
1264	56485.90	-40.43	-12.20
1265	56529.82	-39.76	-12.41
1266	56571.68	-39.06	-12.63
1267	56611.33	-38.34	-12.90
1268	56648.60	-37.59	-13.20
1269	56683.39	-36.82	-13.54
1270	56715.62	-36.02	-13.91
1271	56745.21	-35.21	-14.30
1272	56772.14	-34.27	-15.04
1273	56796.54	-33.03	-16.47
1274	56818.86	-31.54	-18.38
1275	56839.72	-29.79	-20.59
1276	56859.87	-27.77	-22.94
1277	56880.12	-25.48	-25.33
1278	56901.35	-22.94	-27.67
1279	56924.50	-20.15	-29.92
1280	56950.60	-17.13	-32.03

Appendix C. Matlab M-files Used Prior to the Simulation

This appendix contains the listings for all the M-files used before the simulation can be run. These files include the CM flight path generation files and the simulation set up files.

The flight path generation files are:

1. GEN_SAV.M
2. CMGND.M
3. CMCIR.M
4. CMLINE.M

"GEN_SAV.M" is run with an input ASCII file as described in Chapter 3. It calls the other three files, and outputs a ".sav" file containing all the information on the CM's ground track, accelerations, headings and bank angles needed in the simulation.

The simulation setup files are:

1. RUNME.M
2. RUNSIM.M
3. CMCASYS.M
4. CMSYS.M
5. SIMGAIN.M

"RUNME.M" is the top level, menu driven file. It queries the user for the choice of CM profiles and the desired look ahead time. The output of these files is the optimal control gains, CM and CMMCA state matrices, and numerous global constants. At this point, the simulation is ready to be run from inside Simulink.

Some Matlab characters:

```
%    - comment character, remainder of line is ignored
;    - used at end of line to prevent Matlab from printing the results of
       that line (to the screen);
       also ends rows within a matrix
^    - exponentiation
==,<,>,<=,>=     - logical operators
[,] - delineate matrices and vectors
'    - matrix transpose;
       start/end of text string
... - current line is to continued on next line
:    - used to create a vector with default spacing of one;
       e.g., D = 1:4 creates the vector D = [1 2 3 4]
```

```
%                                                                    GEN_SAV.M
%                                                                    24 Jan 93
%                                                                    Randy Nelson
% Top level M file in computing and saving ".sav" files for use
%    by "runsim.m"
%
%  Uses: "cmgnd.m," "spec_den.m," and "time_fuz.m"
%    Currently calls: "cm.dat," "cmshort.dat" and "cmlong.dat"
%
%  Output is a number (three as of now) of ".sav" files containing:
%     n x 5 matrix of CM accelerations (U),
%     n x 4 matrix of CM headings, x-y positions and sign(bank) (cm_hpb),
%     and n x 3 matrix of update, reset and look ahead times (fuzzy_time)
%
% NOTE: if you want to prepare a single data file for use by the simulation,
%      run "cmgnd.m" on the ".dat" file and then run "time_fuz.m"
%      on those results (remember, no arguments to either)

%  ***** list of files to use for possible simulation runs  *****
file_list = [
    '  path1';
    '  path3' ];
    '  path4';
    '  path4_30' ];

%  ***** Define some global variables here (outside of "runsim.m")  *****
global start_step look_ahead fuzzy_time U cm_hpb max_ftut max_reset

start_step = 10;                    % do a psd every 10 seconds
num_freqs = 32;                     % divide spectrum into 32 frequencies
max_ftut  = 10;                     % Max delta ftut
max_reset = max_ftut;               % Max delta reset time <= max delta ftut

for row = 1:size(file_list)
  fnameindex = find(file_list(row,:)~=' ');
  fname = file_list(row,fnameindex);
  [U,cm_hpb] = cmgnd(2,fname)     ;     % use all input files in turn
  fuzzy_time = time_fuz;
  eval(['save ' fname '.sav U cm_hpb fuzzy_time']);
 end; % for row = 1:size(file_list)
clear;

return;
```

```
function [accels,cm_nums] = cmgnd(Flag,fname)
%                                                                    CMGND.M
%  CMMCA flight path generator                                       24 Jan 93
%                                                                    Randy Nelson
%  Top level M file in generation of flight paths.
%    Uses files cmcir.m and cmline.m for arcing and straight line
%    portions of the flight path.
%
%   Input: flag is used to skip portions of "cmgnd.m"
%     Flag = 0 => from user at kbd (do all, don't save output)
%     Flag = 1 => from "pwr_max.m" (skip plots, filename req, cm_nums)
%     Flag = 2 => from "gen_sav.m" (skip plots,filename req)
%     fname is used with "pwr_max.m" and "gen_sav.m"
%
%   Output is a matrix of x and y second derivatives (accels),
%     and n x 4 matrix of CM heading, inertial x & y positions
%      and the sign of the CM bank angle (cm_nums)
%     throughout CM flight time (at one second intervals)
%
%   Format of input data:
%      id   value
%       where
%              / +1 for a Right Turn
%        id is <  -1 for a Left Turn
%              \  0 for a straight segment, and
%        value is # degrees to turn, or
%                 # feet to proceed in straight leg
%
%  Flight begins at origin, heading North at 400 kts.
%   in keeping with (the bizarre) tradition that X is North,
%   Y is East, and Z is down.
%

%  Set up constants and variables
nm = 6076;                                  % conversion factor
maxBank  = 20*pi/180;                       % CM's max bank angle
velocity = 400*nm/3600;                     % CM and CMMCA's speed in ft/sec
radiusCM = velocity^2/(32.2*tan(maxBank));  % CM's nominal turn radius
heading  = 0;                               % initially heading North
position = [0 0];                           % initially at origin
timestep = 1;                               % delta t, in seconds
pathstep = (velocity * timestep);           % feet per time step
pathoffset = 0;                             % correct for path pts not exactly
                                            %    at end of segment
pathplace = 0;                              % position at start of each segment
xy = [];                                    % 2xn array of flt path positions
xy2nd = [];                                 % 2xn array of second derivatives
timepts = [];                               % 1xn array of discrete time pts

%  *****  Get file with CM flight path data  *****
if (Flag == 0)
  fname = input(['What is the name of the CM flight path data file?' 10 ...
                                              [".dat" will be appended] '],'s');
 end; % if (Flag == 0)
disp(['Loading data file ' fname '.dat']);
eval(['load ' fname '.dat']);
fname=eval(eval('fname'));
[segs,DC] = size(fname);                    % segs is # flight path segments

%disp('Working ...')
```

```
for seg = 1:segs                              % one loop for each segment
%  First divide segment into discrete points by dividing the
%     segment path length by the distance flown in delta t seconds.
  if fname(seg,1) == 0,
    seg_length = fname(seg,2);                % length for straight leg
   else
    seg_length = pi/180*radiusCM*fname(seg,2);  % length for turning leg
   end; % if fname(seg,1) == 0
  path_len = seg_length - pathoffset;     % reduce length by offset into seg
  numbstep = fix(path_len/pathstep);      % number of steps in this leg
  round_len = numbstep*pathstep;          % length of integral # of steps
  pathpts = ([0:numbstep]*pathstep)';     % points along path segment; nx1
  pathpts = (pathplace)*ones(pathpts) + pathpts;   % add offset

  if fname(seg,1) == 0                          % a straight leg
     [heading,position,seg_xy,seg_xy2nd] = ...
       cmline(fname(seg,2),heading,position,pathpts,pathoffset);
   else
     [heading,position,seg_xy,seg_xy2nd] = ...
       cmcir(fname(seg,1),fname(seg,2),heading, ...
            position,pathpts,pathoffset,radiusCM,velocity);
   end; % if fname(seg,1) == 0
  xy = [xy; seg_xy];                        % augment xy with new piece
  xy2nd = [xy2nd; seg_xy2nd];               % augment xy2nd with new piece
  timepts = [timepts; pathpts];             % same for timepts

  pathplace = pathstep+pathpts(length(pathpts));  % set new starting pt
  pathoffset = path_len - round_len;              % what's left over
 end; % for seg = 1:segs

%  *****  array of [time, x, y, z, bank angle; same-dot-dot]  *****
accels = [timepts/velocity xy2nd zeros(length(timepts),2)];


% @@@@@@
%  ***** if in "cmgnd.m" from "pwr_max.m" function call, exit here  *****
% @@@@@@
if (Flag == 1)
  return;                              %  accels is all that's needed
 end; % if (Flag == 1)


%  *****  Compute CM heading, inertial position and signum(bank angle) *****
%               at one second intervals throughout flight
vel = diff(xy);                       % velocity approx delta position (in 1 sec)
vel = [675.11 0;  vel];
hdg = rem(2*pi + atan2(vel(:,2),vel(:,1)),2*pi);    % in range [0,2*pi]

% cm bank angle: -1,0 or 1 for left turn, no turn or right turn
w = length(hdg);       % number of time points in CM heading/position matrix
bank = zeros(w,1);
for i = 2:w-1                                       % in middle of matrix
  bank(i) = hdg(i+1,1)-hdg(i-1,1);
 end; % for i = 2:w-1
bank(1) = hdg(3) - hdg(1);                          % at beginning of matrix
bank(w) = hdg(w) - hdg(w-2);                        % at end of matrix
% account for heading change of 360 degrees as CM passes thru North
bank = (abs(bank) > .17) .* sign(bank) * (-2*pi) + bank;
bank = (abs(bank) >= .0017) .* bank;     % any change < .1 degrees is no
change
bank = sign(bank);

cm_nums = [hdg  xy  bank];
```

```
% @@@@@@
%  ***** if in "cmgnd.m" from "gen_sav" function call, just exit here  *****
% @@@@@@
if (Flag == 2)
  return;                                % don't need plots
 end; % if (Flag == 2)


% ************************************************************************
%     Provide plots of Cruise Missile flight path data, if desired:     *
% ************************************************************************

input('Do you want a plot of the second derivatives <Y|N>?','s');
if (ans == 'Y') | (ans == 'y')
  z = max(max(xy2nd)) .* ones(timepts);
  clg;
  disp('Plot of second derivatives of x & y vs time.')
  subplot(211);
    plot(timepts,xy2nd(:,1),':g',timepts,[z -z]);
    title('X Acceleration (inertial) of Cruise Missile')
  subplot(212);
    plot(timepts,xy2nd(:,2),':r',timepts,[z -z]);
    title('Y Acceleration (inertial) of Cruise Missile')
    xlabel('Time from Start (sec)')
    ylabel('          Acceleration (feet/sec/sec)')
 end; % if (ans == Y)

input('Do you want a plot of the flight path <Y|N>?','s');
shg;
if (ans == 'Y') | (ans == 'y')
  clg;
  plot(xy(:,2),xy(:,1),'-w');hold on;
  for jj = 10:10:length(timepts)
    plot(xy(jj,2),xy(jj,1),'*w')
   end;
  plot(xy(1,2),xy(1,1),'*g');
  plot(xy(length(xy),2),xy(length(xy),1),'*r');hold off;
  title('Flight Path of Cruise Missile')
  xlabel('Y distance (feet)')
  ylabel('X distance (feet)')
 end; % if (ans == Y)

return;
```

```
function [finalHDG,finalPOS,seg_xy,seg_xy2nd] = ...
         cmcir(turnDIR,turnANGLE,initHDG,initPOS,pathpts,pOFF,radius,vel)
%                                                                    CMCIR.M
%                                                                    24 Jan 93
%                                                                    Randy Nelson
%
% Purpose:  generate a cubic smoothing spline with initial position,
%    initial heading, degrees to turn and turn direction.
%
%   Inputs: 4 scalars, an n x 1 vector, 3 scalars
%     turnDIR is -1 or +1 for left or right turn,
%     turnANGLE is number of degrees to turn,
%     initHDG is heading of CM at start of turn,
%     initPOS is position of CM at start of turn,
%     pathpts is vector of equally spaced points along the CM's path,
%     pOFF is first point along this segment's path (since discrete points),
%     radius is CM's turn radius,
%     and velocity is CM's speed (constant throughout).
%
%   Outputs: 2 scalars, two n x 1 vectors
%     finalHDG is heading of CM at end of turn,
%     finalHDG is position of CM at end of turn,
%     seg_xy is vector of x and y inertial positions every second,
%     and seg_xy2nd is vector of x and y accelerations at one second intervals.
%
% Note: pathpts changed internally to n+1 points at equal angular intervals,
%     with n = 3 + one for every ten degrees of turn.
%

% set output value and adjust to range [0,360]
finalHDG = rem(initHDG + turnANGLE * turnDIR + 3600,360);

%    Now to parameterize the curve as a pair of functions of s:
%        x=x(s) and y=y(s) w/s the discrete angular step.

%   compute values for circular path parameters

tangle = turnANGLE * pi/180;                    % convert to radians
ihdg   = initHDG   * pi/180;

ctr=radius* [cos(ihdg+pi/2*turnDIR) sin(ihdg+pi/2*turnDIR)] + initPOS;

if turnDIR == -1,                          % left turn
  startangle = -ihdg;                       % using degrees from standard zero
  finalangle = -ihdg + tangle;              %   same convention
 else                                       % right turn
  startangle = pi - ihdg;
  finalangle = startangle - tangle;
end;

N = fix(turnANGLE/10) + 3;     % Number of break points in interval.
i=[0:N];                       % i = 0,1,2,...,N
s = i*tangle/N;                % s is discrete angular step in radians.
w = startangle - s*turnDIR;    % w is step from start to final angle.

%    Discretize x(w) and y(w)
xy = [ctr(1)*ones(w)', ctr(2)*ones(w)']' + radius* [sin(w)' cos(w)']';
finalPOS = [xy(1,N+1) xy(2,N+1)];              % set output value
```

```
%    Now fit spline to x(w) and y(w)

ppx = csapi(w,xy(1,:));        % get pp form of spline for x (1 x n matrix)
ppy = csapi(w,xy(2,:));        % get pp form of spline for y

%  Compute 2nd derivative splines of x(t) and y(t)
pp2nd=[fnder(ppx,2)' fnder(ppy,2)'];          % return an n x 2 matrix

%  Values of points along path converted to angular displacement:

pathpts = (pathpts+(pOFF-pathpts(1))*ones(pathpts))*(1/radius);
pathpts = (startangle*ones(pathpts) - pathpts*turnDIR);

%  Compute positions along the flight path
seg_xy = [fnval(ppx,pathpts')' fnval(ppy,pathpts')'];

%  Compute values of 2nd derivatives along the flight path
seg_xy2nd = (vel^2/radius^2) * ...
            [fnval(pp2nd(:,1)',pathpts')' fnval(pp2nd(:,2)',pathpts')'];

return;
```

```
function [finalHDG,finalPOS,seg_xy,seg_xy2nd] = ...
                        cmline(distance,initHDG,initPOS,pathpts,pOFF)
%                                                                    CMLINE.M
%                                                                    23 Jan 93
%                                                                    Randy Nelson
%
% Purpose:  generate a straight line given the starting point
%    and the line length.
%
%    No default assumptions; need all three values as input:
%     Format is
%       1.  Distance is in feet.
%       2.  initHDG is in compass degrees 0 - 360
%             (i.e. 0 = North, 90 = East, 180 = South and 270 = W).
%       3.  initPOS=[x,y] in feet.
%

%   Now to parameterize the line as a pair of functions
%        of s: x=x(s) and y=y(s) w/s the discrete step.
%

finalHDG = initHDG;              % set output value
initHDG  = initHDG * pi/180;     % convert to radians

%  *****  Compute end point of line  *****
N = 5;                           % Number of break points in interval.
i=[0 1];                         % i = 0,1; start/end of line.

xend = initPOS + distance * [cos(initHDG) sin(initHDG)];
finalPOS = xend;                 % set output value
xy = [initPOS' xend'];           % 2x2 matrix of start/end of line

%    Now fit spline to x(w) and y(w)

ppx = csapi(i,xy(1,:));          % get pp form of spline for x
ppy = csapi(i,xy(2,:));          % get pp form of spline for y

%  Compute 2nd derivative splines of x(t) and y(t):
pp2nd=[fnder(ppx,2)' fnder(ppy,2)'];            % return an n x 2 matrix

%  Values of points along path converted to [0:1] parameter
pathpts1 = (pathpts+(pOFF-pathpts(1))*ones(pathpts));
pathpts2 = pathpts1/max(pathpts1);

%  Compute positions along the flight path
seg_xy = [fnval(ppx,pathpts2')' fnval(ppy,pathpts2')'];

%  Compute values of 2nd derivatives along the flight path
seg_xy2nd = [fnval(pp2nd(:,1)',pathpts')' fnval(pp2nd(:,2)',pathpts')'];

return;
```

```
%                                                                    RUNME.M
%                                                                    18 Dec 92
%                                                                    Randy Nelson
% M File to execute simulation
%  Uses saved missile data in ".sav" files from
%   "path1.sav"  "path3.sav"  "path4.sav"  "path4_30.sav"
%
%  Calls runsim to set simulation parameters and ask about a diary output.

clear;clc;

disp([10 'Welcome to the CMMCA simulation system.' 10])
disp(['You have the choice of eight preset cruise missile flight paths,' ...
      10 'or you may select one of your own.' 10])
disp(['Hachman''s profile 1 (path A, a 180 degree right turn)?       <1>' ...
   10 'Hachman''s profile 3 (path B, 2x 270 degree turns)?           <2>' ...
   10 'Hachman''s profile 4 (path C, a number of turns)?             <3>' ...
   10 'Hachman''s profile 4 (using 30 degrees of bank)?              <4>' ...
   10 'Your choice (you will need to provide a CM flight profile)?  <5>'  ...
   10 '                                          (or Ctrl-C to exit)'])
bad_input = 1;
while bad_input
  fnum=input('Make your choice by selecting a number from 1-9, now:  ','s');
  bad_input = 0;
  if (fnum == '1')
    file = 'path1';
   elseif (fnum == '2')
    file = 'path3';
   elseif (fnum == '3')
    file = 'path4';
   elseif (fnum == '4')
    file = 'path4_30';
   elseif (fnum == '5')
    file = input([10 'Enter the file name containing the flight data.' ...
    10 '(A default extension of ".sav" is always added.)  '],'s');
    if ([file ' '] == ' ') | (exist([file '.sav']) ~= 2)
      disp('You didn''t enter a valid name, back to the beginning ...')
      bad_input = 1;
     end; % if ([file ' '] == ' ') | ...
   else                                              % invalid choice
    bad_input = 1;
    disp([10 'Oops, you''ve selected an invalid choice.  Try again.'])
   end; % if (ans==1)
 end; % while bad_input
disp([10 'Well done; "' file '.sav" is an excellent choice!'])
disp([10 'Loading data file ' file '.sav'])
eval(['load ' file '.sav']);
disp('Loading complete.')
clear bad_input fnum file                % clean up the work space

runsim;

disp([10 'At your leisure, run the simulation.' 10 'It will run for ' ...
      sprintf('%-3.1f seconds.',tfinal)])

return;
```

```
%                                                                    RUNSIM.M
%                                                                    24 Jan 93
%                                                                    Randy Nelson
%   M File to execute simulation; called from "runme.m"
%   Contains most parameters used in simulation and creates
%    diary file on request
%
%   Rcd = True will print out data for diary
%

global Rcd True False vcmca gravity U
global ftut t_reset t_ahead max_ftut max_reset look_ahead
global start_step num_freqs tfinal fuzzy_time
global min_radar_rng max_radar_rng Rng_nominal
global max_left_azim max_right_azim max_left_elev max_right_elev
global Xcm Xcmca accel_limits vcmca_min vcmca_max Kc
global lim_enable setpoint fuz_enable pcxy_max cm_hpb
global azi_now_wt rng_now_wt sim_run_tim sim_start_time

True = 1; False = 0;
sim_start_time = [];                    % stores elapsed time of simulation
sim_run_tim = 0;
Rcd = False ;
nm = 6076;
gravity = 32.2;                         % ft/sec/sec
deg2rad = pi/180;
vcmca = 400*6076/3600;                  % 400 kts (in ft/sec)

input(' What value of look ahead time do you want to use (65,95 or 125)? ');
 t_ahead = ans;

if 0
input(' Output data to diary - <Y/N>? ','s') ;
if (ans == 'Y') | (ans == 'y')
   Rcd = True ;
   diary runsim.dat;
   diary on;
 end
end

disp([10 'There will be a slight delay while the system is set up.' ...
      10 'Please stand by.'])

%      Run CMsys.m to define the state matrices
%          Am,Bm,Cm,Dm
%

cmsys

%      Run CMCAsys.m to define the state matrices
%          Acmca,Bcmca,Ccmca,Dcmca
%

cmcasys

%      Run simgain.m to determine the control system gains (Kc)
%

SampleTime = 1;                         % Discrete sample time
simgain

%
```

```
horz_rng = sqrt((10*6076)^2 - 28000^2);    % for 10nm slant range
%          Initial condition for CM is Xcm
%          Initial condition for CMMCA is Xcmca
Xcm   = [horz_rng  1000  -1000  0   vcmca    0    0    0]' ;
Xcmca = [    0        0  -29000  0   vcmca    0    0    0]' ;
setpoint = Xcm;
%


%
%  ********* SIMULATION PARAMETERS (default values)  ********
%

%  *****  time parameters  *****
tstart  = U(1,1)  ;                % Start time is first element of column 1
tfinal  = U(length(U),1)  ;        % Final time is last element of column 1
ftut = 0;                          % Final time update time
t_reset = 0;                       % Reset time (less than or equal to ftut)
%t_ahead = 0;                       % Look ahead time
max_ftut  = 10;                    % Max delta ftut
max_reset = max_ftut;              % Max delta reset time <= max delta ftut
look_ahead = [65 125 185];         % vector of look ahead times
start_stop = 10;                   % step thru U at this interval
num_freqs = 32;                    % number of freq intervals looked at

% This a matrix of max psd values for a (very) long random CM profile
%  the values were computed in "pwr_max.m" and manually imported here
pcxy_max = [12.34 23.45 34.56;     % max value of 'x' psd
            45.67 56.78 67.89;     % max value of 'y' psd
            12.34 23.45 34.56;     % max value of 'x' 95% confidence interval
            45.67 56.78 67.89 ];   % max value of 'y' 95% confidence interval

%  *****  simulation parameters  *****
tol     = 1e-2 ;
minstep = 1e-0 ;
maxstep = 1e+0 ;
max_data_pts = 5000;                    % max rows in output to workspace

%
%  *****  CMMCA PARAMETERS  *****
%

%  ***** radar parameters
min_radar_rng =  5 * nm;
max_radar_rng = 15 * nm;
Rng_nominal   = 10 * nm;
max_left_azim  =-60 * deg2rad;
max_right_azim = 60 * deg2rad;
max_down_elev  =-60 * deg2rad;
max_up_elev    = 60 * deg2rad;

%  *****  velocity limits  *****
vcmca_min = 360 *nm/3600;
vcmca_max = 480 *nm/3600;

%  *****  Acceleration limits in the body axis  *****
%     +/- maximum speed change in feet/s/s,
%     + sideslip g-limit (not currently used),
%     negative g-limit (positive number is OK if < 1 g),
%     bank angle limit in degrees,
%     roll rate limit in degrees/s,
%     and bank angle acceleration limit in degrees/s/s.
```

```
accel_limits = [  0      0    ...                  % -+ speed accel limits
                  0           ...                  % side slip limit
            0.75*gravity   2.0*gravity  ...        % min and max g limits
                 30*deg2rad    ...                  % max bank angle
                  8*deg2rad];                       % max roll accel

%  ***** DIARY DOINGS DOWN HERE  *****
if Rcd == True
  disp([10 10 'From runsim.m: ' 10]);
  disp(['CM initial position     = ' sprintf('%8.3f',Xcm(1)) '   ' ...
     sprintf('%8.3f',Xcm(2)) '   ' ...
     sprintf(' %8.3f',Xcm(3)) '   ' sprintf('%8.3f',Xcm(4))]);
  disp(['                          ' sprintf('%8.3f',Xcm(5)) '   ' ...
     sprintf('%8.3f',Xcm(6)) '     ' sprintf('%8.3f',Xcm(7)) '   ' ...
     sprintf('%8.3f\n',Xcm(8))]);
  disp(['CMMCA initial position  = ' sprintf('%8.3f',Xcmca(1)) '   ' ...
     sprintf('%8.3f',Xcmca(2)) '   ' sprintf('%8.3f',Xcmca(3)) '   ' ...
     sprintf('%8.3f',Xcmca(4))]);
  disp(['                          ' sprintf('%8.3f',Xcmca(5)) '   ' ...
     sprintf('%8.3f',Xcmca(6)) '     ' sprintf('%8.3f',Xcmca(7)) '   ' ...
     sprintf('%8.3f\n',Xcmca(8))]);
  disp(['tstart =  ' sprintf('%8.3f',tstart)]);
  disp(['tfinal =  ' sprintf('%8.3f',tfinal)]);
  disp(['tol =       ' sprintf('%8.3e',tol)]);
  disp(['minstep =   ' sprintf('%8.3e',minstep)]);
  disp(['maxstep =   ' sprintf('%8.3e',maxstep)]);
  disp(['max_data_pts = ' sprintf('%8.3f',max_data_pts)]) ;
  al=accel_limits;
  disp(['Airspeed change limits          = ' sprintf(' %8.3f    %8.3f' ...
        ,al(1),al(2)) '  feet/s/s']);
  disp(['Sideslip g-limit                = ' ...
        sprintf(' %8.3f',al(3)/gravity) '  g''s']);
  disp(['Minimum/maximum g-limits        = ' ...
        sprintf(' %8.3f    /  %5.3f',al(4)/gravity,al(7)/gravity) '  g''s']);
  disp(['Bank angle and bank accel limit = ' ...
        sprintf(' %8.3f degs  %5.3f',al(5)*180/pi,al(6)*180/pi) ...
        '  degrees/s/s']);
  disp(['CMMCA airspeed limits:  min A/S  = ' ...
     sprintf(' %3.0f kts (%7.3f feet/sec)',vcmca_min*3600/nm,vcmca_min) ...
     10 '                          max A/S  = ' ...
     sprintf(' %3.0f kts (%7.3f feet/sec)',vcmca_max*3600/nm,vcmca_max)]);
  diary off;
end

return;
```

```
%                                                          CMCASYS.M
%                                                          1 Dec 92
%                                                          D. Caughlin
%  M File to initialize the cmmca system matrices
%
%   Variables Acmca, Bcmca are the basic system matrices
%             Ccmca, Dcmca are the output matrices
%
%                                    T
%   x    = [x1 x2 x3 x4 x5 x6 x7 x8]
%                  x1 = x position
%                  x2 = y position
%                  x3 = z position
%                  x4 = bank angle
%                  x5 = x velocity
%                  x6 = y velocity
%                  x7 = z velocity
%                  x8 = bank angle rate
%
%                     T
%   u    = [u1 u2 u3 u4]
%              u1 = x acceleration
%              u2 = y acceleration
%              u3 = z acceleration
%              u4 = roll acceleration

global Acmca Bcmca Ccmca Dcmca

w = .2;                                % w is really the parameter "omega"

Acmca = ...
   [ 0     0     0     0     1     0     0     0;
     0     0     0     0     0     1     0     0;
     0     0     0     0     0     0     1     0;
     0     0     0     0     0     0     0     1;
     0     0     0     0     0     0     0     0;
     0     0     0     0     0     0     0     0;
     0     0     0     0     0     0     0     0;
     0     0     0     0     0     0     0    -w];

Bcmca = ...
    [0     0     0     0;
     0     0     0     0;
     0     0     0     0;
     0     0     0     0;
     1     0     0     0;
     0     1     0     0;
     0     0     1     0;
     0     0     0     w];

Ccmca = ...
    [1     0     0     0     0     0     0     0;
     0     1     0     0     0     0     0     0;
     0     0     1     0     0     0     0     0;
     0     0     0     1     0     0     0     0;
     0     0     0     0     1     0     0     0;
     0     0     0     0     0     1     0     0;
     0     0     0     0     0     0     1     0;
     0     0     0     0     0     0     0     1];
```

```
Dcmca = ...
    [0     0     0     0;
     0     0     0     0;
     0     0     0     0;
     0     0     0     0;
     0     0     0     0;
     0     0     0     0;
     0     0     0     0;
     0     0     0     0];

if Rcd == True
  sp = '     ';
  disp([10 10 'From cmcasys.m: ' 10])
  disp([10 'Acmca =      ' 10 '        '])
  for l = 1:8
   disp([sp sprintf('%5.2f',Acmca(l,1)) sp sprintf('%5.2f',Acmca(l,2)) sp ...
        sprintf('%5.2f',Acmca(l,3)) sp sprintf('%5.2f',Acmca(l,4)) sp ...
        sprintf('%5.2f',Acmca(l,5)) sp sprintf('%5.2f',Acmca(l,6)) sp ...
        sprintf('%5.2f',Acmca(l,7)) sp sprintf('%5.2f',Acmca(l,8))])
   end;
  disp([10 'Bcmca =      ' 10 '        '])
  for l = 1:8
   disp([sp sprintf('%5.2f',Bcmca(l,1)) sp sprintf('%5.2f',Bcmca(l,2)) sp ...
        sprintf('%5.2f',Bcmca(l,3)) sp sprintf('%5.2f',Bcmca(l,4))])
   end;
  disp([10 'Ccmca =      ' 10 '        '])
  for l = 1:8
   disp([sp sprintf('%5.2f',Ccmca(l,1)) sp sprintf('%5.2f',Ccmca(l,2)) sp ...
        sprintf('%5.2f',Ccmca(l,3)) sp sprintf('%5.2f',Ccmca(l,4)) sp ...
        sprintf('%5.2f',Ccmca(l,5)) sp sprintf('%5.2f',Ccmca(l,6)) sp ...
        sprintf('%5.2f',Ccmca(l,7)) sp sprintf('%5.2f',Ccmca(l,8))])
   end;
  disp([10 'Dcmca =      ' 10 '        '])
  for l = 1:8
   disp([sp sprintf('%5.2f',Dcmca(l,1)) sp sprintf('%5.2f',Dcmca(l,2)) sp ...
        sprintf('%5.2f',Dcmca(l,3)) sp sprintf('%5.2f',Dcmca(l,4))])
   end;
 end

return
```

```
%                                                          CMSYS.M
%                                                          1 Dec 92
%                                                          D. Caughlin
%  M File to initialize the cruise missile system matrices
%
%   Variables Acm, Bcm are the basic system matrices
%             Ccm, Dcm are the output matrices
%
%                                    T
%   x    = [x1 x2 x3 x4 x5 x6 x7 x8]
%                    x1 = x position
%                    x2 = y position
%                    x3 = z position
%                    x4 = bank angle
%                    x5 = x velocity
%                    x6 = y velocity
%                    x7 = z velocity
%                    x8 = bank angle rate
%
%                      T
%   u    = [u1 u2 u3 u4]
%                u1 = x acceleration
%                u2 = y acceleration
%                u3 = z acceleration
%                u4 = roll acceleration
%
% NOTE: the roll rate must be changed

w = .1;

global Acm Bcm Ccm Dcm

Acm=[0 0 0 0 1 0 0 0;
     0 0 0 0 0 1 0 0;
     0 0 0 0 0 0 1 0;
     0 0 0 0 0 0 0 1;
     0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 -w];

Bcm=[0 0 0 0;
     0 0 0 0;
     0 0 0 0;
     0 0 0 0;
     1 0 0 0;
     0 1 0 0;
     0 0 1 0;
     0 0 0 w];

Ccm=[1 0 0 0 0 0 0 0;
     0 1 0 0 0 0 0 0;
     0 0 1 0 0 0 0 0;
     0 0 0 1 0 0 0 0;
     0 0 0 0 1 0 0 0;
     0 0 0 0 0 1 0 0;
     0 0 0 0 0 0 1 0;
     0 0 0 0 0 0 0 1];
```

```
Dcm=[0 0 0 0;
     0 0 0 0;
     0 0 0 0;
     0 0 0 0;
     0 0 0 0;
     0 0 0 0;
     0 0 0 0;
     0 0 0 0];

if Rcd == True
  sp = '     ';
  disp([10 10 'From cmsys.m: ',10])
  disp([10 'Acm =      ' 10 '        '])
  for l = 1:8
  disp([sp sprintf('%5.2f',Acm(1,1)) sp sprintf('%5.2f',Acm(1,2)) sp ...
        sprintf('%5.2f',Acm(1,3)) sp sprintf('%5.2f',Acm(1,4)) sp ...
        sprintf('%5.2f',Acm(1,5)) sp sprintf('%5.2f',Acm(1,6)) sp ...
        sprintf('%5.2f',Acm(1,7)) sp sprintf('%5.2f',Acm(1,8))])
   end;
  disp([10 'Bcm =      ' 10 '        '])
  for l = 1:8
  disp([sp sprintf('%5.2f',Bcm(1,1)) sp sprintf('%5.2f',Bcm(1,2)) sp ...
        sprintf('%5.2f',Bcm(1,3)) sp sprintf('%5.2f',Bcm(1,4))])
   end;
  disp([10 'Ccm =      ' 10 '        '])
  for l = 1:8
  disp([sp sprintf('%5.2f',Ccm(1,1)) sp sprintf('%5.2f',Ccm(1,2)) sp ...
        sprintf('%5.2f',Ccm(1,3)) sp sprintf('%5.2f',Ccm(1,4)) sp ...
        sprintf('%5.2f',Ccm(1,5)) sp sprintf('%5.2f',Ccm(1,6)) sp ...
        sprintf('%5.2f',Ccm(1,7)) sp sprintf('%5.2f',Ccm(1,8))])
   end;
  disp([10 'Dcm =      ' 10 '        '])
  for l = 1:8
  disp([sp sprintf('%5.2f',Dcm(1,1)) sp sprintf('%5.2f',Dcm(1,2)) sp ...
        sprintf('%5.2f',Dcm(1,3)) sp sprintf('%5.2f',Dcm(1,4))])
   end;
end; % if Rcd == True

return
```

```
%                                                              SIMGAIN.M
%                                                              1 Dec 92
%                                                              D. Caughlin
%                                                                revised
%                                                              9 Jan 93
%                                                              Randy Nelson
%  M File used to compute the Steady State LQG regulator gains
%
%  Uses Acmca,Bcmca,Bcmca,Dcmca from the workplace - 8 state system
%
%   x =
%         x1 = x position
%         x2 = y position
%         x3 = z position
%         x4 = bank angle
%         x5 = x velocity
%         x6 = y velocity
%         x7 = z velocity
%         x8 = bank angle rate
%
%   u =
%         u1 = x acceleration
%         u2 = y acceleration
%         u3 = z acceleration
%         u4 = roll acceleration
%
%  Uses weighting functions Q & R from this file

%
%  Design an LQG regulator
%
%  NOTE: LQRD() will not work with this structure
%        Reformulate the system matrices into
%
%   x =
%         x1 = x position
%         x2 = x velocity
%         x3 = y position
%         x4 = y velocity
%         x5 = z position
%         x6 = z velocity
%         x7 = bank angle
%         x8 = bank angle rate
%

Acmca1 = Acmca([1 5 2 6 3 7 4 8],:);
Acmca1 = Acmca1(:,[1 5 2 6 3 7 4 8]);

Bcmca1 = Bcmca([1 5 2 6 3 7 4 8],:);


   Qc = [ 10 0  0  0  0  0  0  0;
          0  0  0  0  0  0  0  0;
          0  0  10 0  0  0  0  0;
          0  0  0  0  0  0  0  0;
          0  0  0  0  10 0  0  0;
          0  0  0  0  0  0  0  0;
          0  0  0  0  0  0  5  0;
          0  0  0  0  0  0  0  0  ];
```

```
    Rc = [ 1 0 0 0;
           0 1 0 0;
           0 0 1 0;
           0 0 0 1  ];

    [Kc,Sc] = lqrd(Acmcal,Bcmcal,Qc,Rc,SampleTime);

%
% Reset control and Ricatti solution for original states
%

Kc = Kc(:,[1 3 5 7 2 4 6 8]);

Sc = Sc(:,[1 3 5 7 2 4 6 8]);
Sc = Sc([1 3 5 7 2 4 6 8],:);

clear Acmcal Bcmcal                          % clean up the workspace

if Rcd == True
  sp = '     ';
  disp([10 10 'From simgain.m: ' 10])
  disp([10 'Qc =      ' 10 '       '])
  for l = 1:8
  disp([sp sprintf('%5.2f',Qc(1,1)) sp sprintf('%5.2f',Qc(1,2)) sp ...
        sprintf('%5.2f',Qc(1,3)) sp sprintf('%5.2f',Qc(1,4)) sp ...
        sprintf('%5.2f',Qc(1,5)) sp sprintf('%5.2f',Qc(1,6)) sp ...
        sprintf('%5.2f',Qc(1,7)) sp sprintf('%5.2f',Qc(1,8))])
   end;
  disp([10 'Rc =      ' 10 '       '])
  for l = 1:4
  disp([sp sprintf('%5.2f',Rc(1,1)) sp sprintf('%5.2f',Rc(1,2)) sp ...
        sprintf('%5.2f',Rc(1,3)) sp sprintf('%5.2f',Rc(1,4))])
   end;
   disp('  ')
   disp('    LQG Regulator Design with restructured system matrices:')
   disp('  ')
   disp('    ------------------------------------------------------------')
   disp('       [K,S] = lqrd(Acmcal,Bcmcal,Qc,Rc)  % Computing the regulator gain K')
   disp('                                          % and soln to Ricatti eqn S')
   disp('    ------------------------------------------------------------')
  disp([10 'Kc =      ' 10 '       '])
  for l = 1:4
  disp([sp sprintf('%5.2f',Kc(1,1)) sp sprintf('%5.2f',Kc(1,2)) sp ...
        sprintf('%5.2f',Kc(1,3)) sp sprintf('%5.2f',Kc(1,4)) sp ...
        sprintf('%5.2f',Kc(1,5)) sp sprintf('%5.2f',Kc(1,6)) sp ...
        sprintf('%5.2f',Kc(1,7)) sp sprintf('%5.2f',Kc(1,8))])
   end;
  disp([10 'Sc =      ' 10 '       '])
  for l = 1:8
  disp([sp sprintf('%5.2f',Sc(1,1)) sp sprintf('%5.2f',Sc(1,2)) sp ...
        sprintf('%5.2f',Sc(1,3)) sp sprintf('%5.2f',Sc(1,4)) sp ...
        sprintf('%5.2f',Sc(1,5)) sp sprintf('%5.2f',Sc(1,6)) sp ...
        sprintf('%5.2f',Sc(1,7)) sp sprintf('%5.2f',Sc(1,8))])
   end;
end; % if Rcd == True

return;
```

Appendix D: Matlab M-files Called by the Simulation

This appendix contains the listings for all the M-files used during the actual run of the simulation. See Appendix C for more details on Matlab.

```
function inert = body2inert(u)
%                                                          BODY2INERT.M
%                                                          22 Nov 92
%                                                          Randy Nelson
% This accepts a 6 x 1 vector as input:
%  The body controls (accelerations), a 3 x 1 vector,
%   and the three euler angles (psi, theta and phi).
%
% Output is the inertial axis accelerations (3 x 1 vector)

psi   = u(4);
theta = u(5);
phi   = u(6);

%  ******************    Compute transformation matrix D  ************

D = zeros(3,3);                          % speeds up memory allocation a bit
D(1,1) = cos(theta)*cos(psi);
D(1,2) = cos(theta)*sin(psi);
D(1,3) = -sin(theta);
D(2,1) = sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi);
D(2,2) = sin(phi)*sin(theta)*sin(psi) + cos(phi)*cos(psi);
D(2,3) = sin(phi)*cos(theta);
D(3,1) = cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi);
D(3,2) = cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi);
D(3,3) = cos(phi)*cos(theta);

inert = D'*[u(1) u(2) u(3)]';
```

```
function [next_hdg,next_azi] = ...
                    cm_next(t_next,cmmca_state,cmmca_hdg)
%                                                                    CM_NEXT.M
%                                                                    23 Jan 93
%                                                                    Randy Nelson
%
% Computes CM heading and azimuth at future time
%
%  Inputs: the future time (t_next),
%    the current CMMCA state (for inertial position), an 8 x 1 vector,
%    and the current CMMCA heading (cmmca_hdg)
%   Uses: matrix of CM headings and positions (cm_hpb).
%
%  Output: CM heading and azimuth from CMMCA present position
%    to the CM's future position
%
                                          % matrix is in one second intervals
t_next = round(min(t_next,length(cm_hpb))); %  & stay within simulation time

next_hdg = cm_hpb(t_next,1);              % CM heading at t_next
next_pos = cm_hpb(t_next,2:3)'+Xcm(1:2);  % inertial CM position at t_next

% convert CM future position relative to CMMCA present position to an azimuth

%  ******************    Compute transformation matrix d  ************
%   derived from D with theta = 0 & phi = 0
d = zeros(2,2);                           % allocate space to speed things up
d(1,1) = cos(cmmca_hdg);
d(1,2) = sin(cmmca_hdg);
%D(1,3) = 0;
d(2,1) = -sin(cmmca_hdg);
d(2,2) = cos(cmmca_hdg);
%D(2,3) = 0;
%D(3,1) = 0;
%D(3,2) = 0;
%D(3,3) = 1;

% transform to body axis
rel_pos = next_pos - cmmca_state(1:2);   % CMMCA current to CM future position
body_pos = d * rel_pos;

%  Compute azimuth
next_azi = atan2(body_pos(2),body_pos(1));       % next_azi in [-pi,pi]

return;
```

```
function phicmd = cmdbank(u)
%                                                              CMDBANK.M
%                                                              4 Dec 92
%                                                              Randy Nelson
% This accepts two column vectors as input:
%  the euler angles psi, theta, and phi (yaw, pitch and bank)
%  and the inertial controls x, y and z (and phi, but not used).
%
% Output is the commanded bank angle phicmd.
%

psi   = u(1);
theta = u(2);
phi   = u(3);

%  ******************    Compute transformation matrix D  ************

D = zeros(3,3);                          % speeds up memory allocation a bit
D(1,1) = cos(theta)*cos(psi);
D(1,2) = cos(theta)*sin(psi);
D(1,3) = -sin(theta);
D(2,1) = sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi);
D(2,2) = sin(phi)*sin(theta)*sin(psi) + cos(phi)*cos(psi);
D(2,3) = sin(phi)*cos(theta);
D(3,1) = cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi);
D(3,2) = cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi);
D(3,3) = cos(phi)*cos(theta);

%  ******************   Transform inertial accel's to body ************

Lb = D*[u(4) u(5) u(6)]';              % 3 x 1 vector of body accel's

%  *****  Compute commanded bank angle  *****

if (abs(Lb(2)) < 1e-1)                    % make sure tiny values don't
  Lb(2) = 0;                              %  confuse the issue
 end; % if (abs(Lb(2)) < 1e-1)
if (abs(Lb(3)) < 1e-1)
  Lb(3) = 0;
 end; % if (abs(Lb(3)) < 1e-1)

phibody = atan2(Lb(2), -Lb(3));           % commanded bank angle in body axis
phicmd = phi + phibody;                   % add to CMMCA bank to get net
if (phicmd < -pi)
  phicmd = phicmd + 2*pi;                 % make sure angle in [-pi,pi]
 elseif (phicmd > pi)
  phicmd = phicmd - 2*pi;
 end; % if (phicmd < -pi)

return;
```

```
function angles = eulerang(u)
%                                                                    EULERANG.M
%                                                                    15 Dec 92
%                                                                    Randy Nelson
%
% This accepts an 8 x 1 column vector as input; the state vector
%   in the inertial coordinate F.O.R.
%   (x,y,z and phi, and their derivatives).
%
% Output is the euler angles psi, theta, and phi (yaw, pitch and bank).
%  Limits on angles:
%         -180 <=  psi  <= +180
%          -90 <= theta <=  +90
%         -180 <=  phi  <= +180
%

vcmca = norm(u(5:7)) + 1;                    % add 1 so velocity is never zero

%  ************************     Compute yaw angle psi  *************
%    same as heading since inertial X is North.

psi = pi/2 - atan2(u(5),u(6));                % 4-quad arctan(vy/vx)
if psi < 0
  psi = psi + 2*pi;
      if debugg == 1, psi=psi*180/pi, end;
 end; %if psi < 0

% ************************   Compute pitch angle theta  *************

%theta = acos(u(7)/vcmca) - pi/2;
theta = -asin(u(7)/vcmca);

% ************************    Compute bank angle phi  *************

phi = u(4);     % straight out of CMMCA state vector

angles = [psi theta phi];

return;
```

```
function out = gimbal(u)
%                                                                    GIMBAL.M
%                                                                    7 Jan 93
%                                                                    Randy Nelson
% Inputs:
%  the CM and the CMMCA state vectors (inertial),
%   and the euler angles (psi, theta and phi).
%
%  Output:
%   the CM and the CMMCA state vectors (inertial),
%    the euler angles (psi, theta and phi),
%    the range from CMMCA to CM and gimbal angles (psi_g, theta_g),

%  *****  Attach names to inputs  *****
cm_state    = u(1:8);
cmmca_state = u(9:16);
psi         = u(17);
theta       = u(18);
phi         = u(19);

%  ******************    Compute transformation matrix D  ************
D = zeros(3,3);                          % speeds up memory allocation a bit
D(1,1) = cos(theta)*cos(psi);
D(1,2) = cos(theta)*sin(psi);
D(1,3) = -sin(theta);
D(2,1) = sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi);
D(2,2) = sin(phi)*sin(theta)*sin(psi) + cos(phi)*cos(psi);
D(2,3) = sin(phi)*cos(theta);
D(3,1) = cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi);
D(3,2) = cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi);
D(3,3) = cos(phi)*cos(theta);

d_i = cm_state(1:3)-cmmca_state(1:3);      % inertial vector from CMMCA to CM
d_b = D * d_i;                             % convert to body axis

%  ******************    Compute radar range  ************
range = norm(d_i);

%  ******************    Compute azimuth (gimbal psi)  ***************

psi_g = atan2(d_b(2),d_b(1));                % psi_g in [-pi,pi]

%  ******************    Compute pitch (gimbal theta)  ***************

theta_g = -asin(d_b(3)/range);

gim = [range psi_g theta_g];

out = [cm_state; cmmca_state; psi; theta; phi; gim'];

return;
```

```
function  [s_now_azi,s_now_hca,s_rng,s_next_azi,s_next_hca] = ...
                hrd2s(now_azi,now_hca,rng,next_azi,next_hca);
%                                                                HRD2S.M
%                                                                6 Feb 93
%                                                                Randy Nelson
% This file takes the inputs and maps them into the range [-5,5]
%   (if out of [-5,5], "s2fuz.m" will treat as if at end of range)
%
%  Inputs: current radar azimuth to CM (now_azi),
%     current CM-CMMCA heading crossing angle (now_hca),
%     current range to CM,
%     and future values of radar azimuth and HCA (next_azi & next_hca)
%
%  Output is the scaled levels for present radar azimuth and HCA, range,
%   and future (at t_ahead) radar azimuth and HCA.

%       *****  scale the variables  *****
  s_now_azi  = (now_azi * 5 * 2 / (max_right_azim-max_left_azim));
  s_now_hca  = (now_hca * 5 / (3*pi/4));
  s_rng      = (rng-Rng_nominal) * 5 * 2 / (max_radar_rng - min_radar_rng);
  s_next_azi = (next_azi * 5 * 2 / (max_right_azim-max_left_azim));
  s_next_hca = (next_hca * 5 / pi);

return;
```

```
function [cmd_bank,speed_chg] = ...
        log_bank(now_azi,now_hca,rng,next_azi,next_hca,cm_phi)
%                                                                LOG_BANK.M
%                                                                8 Feb 93
%                                                                Randy Nelson
%  Inputs: now_azi, now_hca, rng, next_azi, next_hca, cm_phi
%   Uses: CMMCA_max_bank_angle (accel_limits(6)), t_ahead
%         "s2fuz.m"
%
%  Outputs: CMMCA bank angle and speed change commanded
%    for compute the next set point
%

% Given two sets A and B, and their membership functions m1 and m2,
%  define union, intersection and complement of the sets:
%
%  Intersection corresponds to "AND:"  m(A AND B) = min([m1(x) m2(x)])
%  Union corresponds to "OR:"          m(A OR B)  = max([m1(x) m2(x)])
%  Complement corresponds to "NOT:"    m(NOT A)   = 1 - m1(x)
%
% Output depends on where centroid of logic rules falls in output's fuzzy set.
%

%  ***********        Sets:        *************
%  "A" corresponds to present radar gimbal azimuth (now_azi)
%  "B" corresponds to present CM-to-CMMCA heading crossing angle (now_hca)
%  "C" corresponds to radar range (rng)
%  "D" corresponds to CM bank angle (cm_phi)
%  "E" corresponds to future radar gimbal azimuth at t_ahead (next_azi)
%  "F" corresponds to future CM-to-CMMCA HCA at t_ahead (next_hca)

%  "G" is interim bank angle output from now_azi and now_hca
%  "H" is interim bank angle output from range and CM bank direction

%  "P" corresponds to output bank at t_now  \
%  "Q" corresponds to output bank at t_next  \__ to be converted
%  "R" corresponds to net output bank        /    to set point
%  "S" corresponds to output speed          /

speed_chg = 0;            % don't implement speed changes

%  *****  Make appropriate function calls to eventually get  *****
%              set membership levels for input variables

% Map real variables to scaled integer values in [-5,5]

[s_now_azi,s_now_hca,s_rng,s_next_azi,s_next_hca] = hrd2s ...
                (now_azi,now_hca,rng,next_azi,next_hca);

% Convert scaled values to fuzzy set levels
[ALP,ASP,AZR,ASN,ALN] = s2fuz(s_now_azi);     % present CM-to-CMMCA HCA
[BLP,BSP,BZR,BSN,BLN] = s2fuz(s_now_hca);     % present radar azimuth
[CLP,CSP,CZR,CSN,CLN] = s2fuz(s_rng);         % radar range

[ELP,ESP,EZR,ESN,ELN] = s2fuz(s_next_azi);    % future CM-to-CMMCA HCA
[FLP,FSP,FZR,FSN,FLN] = s2fuz(s_next_hca);    % future radar azimuth
```

```
DPO = 0;                                        % CM bank angle
DZE = 0;
DNE = 0;
if (cm_phi > 0)
  DPO = 1;
 elseif (cm_phi < 0)
  DNE = 1;
 else
  DZE = 1;
 end; % if (cm_phi > 0)

% Define output variables at the center of each trapezoid

LP =  5;
SP =  2.5;
ZR =  0;
SN = -2.5;
LN = -5;

%  ***********       RULES:       ************

%  ***************************************************************************
%         t_now logic for computing bank to next set point                   *
%                          based on                                          *
%          1) radar gimbal azimuth and CM to CMMCA HCA                       *
%          2) range, CM bank angle direction and CM to CMMCA HCA             *
%  ***************************************************************************

%  *****  Azimuth and HCA logic for bank  ****
ab_bank = [
ALP BLP   LP;          % 1a.  If A is LP   AND   B is LP      THEN  P is LP
ALP BSP   LP;          % 1b.  If A is LP   AND   B is SP      THEN  P is LP
ALP BZR   LP;          % 1c.  If A is LP   AND   B is ZR      THEN  P is SP
ALP BSN   SP;          % 1d.  If A is LP   AND   B is SN      THEN  P is SP
ALP BLN   SP;          % 1e.  If A is LP   AND   B is LN      THEN  P is SP

ASP BLP   SP;          % 2a.  If A is SP   AND   B is LP      THEN  P is LP
ASP BSP   SP;          % 2b.  If A is SP   AND   B is SP      THEN  P is SP
ASP BZR   SP;          % 2c.  If A is SP   AND   B is ZR      THEN  P is ZR
ASP BSN   ZR;          % 2d.  If A is SP   AND   B is SN      THEN  P is ZR
ASP BLN   SN;          % 2e.  If A is SP   AND   B is LN      THEN  P is SN

AZR BLP   SP;          % 3a.  If A is ZR   AND   B is LP      THEN  P is LP
AZR BSP   SP;          % 3b.  If A is ZR   AND   B is SP      THEN  P is SP
AZR BZR   ZR;          % 3c.  If A is ZR   AND   B is ZR      THEN  P is ZR
AZR BSN   SN;          % 3d.  If A is ZR   AND   B is SN      THEN  P is SN
AZR BLN   SN;          % 3e.  If A is ZR   AND   B is LN      THEN  P is LN

ASN BLP   SP;          % 4a.  If A is SN   AND   B is LP      THEN  P is SP
ASN BSP   ZR;          % 4b.  If A is SN   AND   B is SP      THEN  P is ZR
ASN BZR   SN;          % 4c.  If A is SN   AND   B is ZR      THEN  P is ZR
ASN BSN   SN;          % 4d.  If A is SN   AND   B is SN      THEN  P is SN
ASN BLN   SN;          % 4e.  If A is SN   AND   B is LN      THEN  P is LN

ALN BLP   SN;          % 5a.  If A is LN   AND   B is LP      THEN  P is SN
ALN BSP   SN;          % 5b.  If A is LN   AND   B is SP      THEN  P is SN
ALN BZR   LN;          % 5c.  If A is LN   AND   B is ZR      THEN  P is SN
ALN BSN   LN;          % 5d.  If A is LN   AND   B is SN      THEN  P is LN
ALN BLN   LN ];        % 5e.  If A is LN   AND   B is LN      THEN  P is LN
```

```
%  *****  Radar range logic for bank  ****
% need to know which way to turn for closure when CM not turning
% get that info from sign(now_azi):
d = sign(now_azi);

cd_bank = [
CLP DPO   LP;          %  6a. If C is LP   AND   D is POS     THEN  P is LP
CLP DZE   LP*d;        %  6b. If C is LP   AND   D is ZERO    THEN  P is S
CLP DNE   LN;          %  6c. If C is LP   AND   D is NEG     THEN  P is LN

CSP DPO   SP;          %  7a. If C is SP   AND   D is POS     THEN  P is SP
CSP DZE   SP*d;        %  7b. If C is SP   AND   D is ZERO    THEN  P is ZR
CSP DNE   SN;          %  7c. If C is SP   AND   D is NEG     THEN  P is SN

CZR DPO   ZR;          %  8a. If C is ZR   AND   D is POS     THEN  P is ZR
CZR DZE   ZR;          %  8b. If C is ZR   AND   D is ZERO    THEN  P is ZR
CZR DNE   ZR;          %  8c. If C is ZR   AND   D is NEG     THEN  P is ZR

CSN DPO   SN;          %  9a. If C is SN   AND   D is POS     THEN  P is SN
CSN DZE   SN*d;        %  9b. If C is SN   AND   D is ZERO    THEN  P is ZR
CSN DNE   SP;          %  9c. If C is SN   AND   D is NEG     THEN  P is SP

CLN DPO   LN;          % 10a. If C is LN   AND   D is POS     THEN  P is LN
CLN DZE   LN*d;        % 10b. If C is LN   AND   D is ZERO    THEN  P is S
CLN DNE   LP ];        % 10c. If C is LN   AND   D is NEG     THEN  P is LP

% fuzzy logic computations for now_bank components:

% append column for logical AND of now_azi and now_hca for bank angle
ab_bank(:,4) = min(ab_bank(:,1),ab_bank(:,2));
ab_bank(:,5) = ab_bank(:,3).*ab_bank(:,4);
now_azi_hca = sum(ab_bank(:,5))/sum(ab_bank(:,4));  % scaled value from azimuth

% append column for logical AND of rng and cm_phi for bank angle
cd_bank(:,4) = min(cd_bank(:,1),cd_bank(:,2));
cd_bank(:,5) = cd_bank(:,3).*cd_bank(:,4);
rng_cmbank = sum(cd_bank(:,5))/sum(cd_bank(:,4));     % scaled value from range
```

```
%  ************************************************************************
%    now_bank logic to combine fuzzy bank logic derived from             *
%              current azimuth, HCA  and range                           *
%  ************************************************************************

% Convert scaled t_now output bank values to fuzzy set levels
[GLP,GSP,GZR,GSN,GLN] = s2fuz(now_azi_hca);     % present CM-to-CMMCA HCA
[HLP,HSP,HZR,HSN,HLN] = s2fuz(rng_cmbank);      % present radar azimuth

now_bank = [
GLP HLP   LP;          % 11a.  If G is LP   AND  H  is LP      THEN  P is LP
GLP HSP   LP;          % 11b.  If G is LP   AND  H  is SP      THEN  P is LP
GLP HZR   LP;          % 11c.  If G is LP   AND  H  is ZR      THEN  P is LP
GLP HSN   LP;          % 11d.  If G is LP   AND  H  is SN      THEN  P is LP
GLP HLN   LP;          % 11e.  If G is LP   AND  H  is LN      THEN  P is LP

GSP HLP   LP;          % 12a.  If G is SP   AND  H  is LP      THEN  P is SP
GSP HSP   SP;          % 12b.  If G is SP   AND  H  is SP      THEN  P is SP
GSP HZR   SP;          % 12c.  If G is SP   AND  H  is ZR      THEN  P is SP
GSP HSN   ZR;          % 12d.  If G is SP   AND  H  is SN      THEN  P is SP
GSP HLN   SN;          % 12e.  If G is SP   AND  H  is LN      THEN  P is ZR

GZR HLP   LP;          % 13a.  If G is ZR   AND  H  is LP      THEN  P is SP
GZR HSP   SP;          % 13b.  If G is ZR   AND  H  is SP      THEN  P is ZR
GZR HZR   ZR;          % 13c.  If G is ZR   AND  H  is ZR      THEN  P is ZR
GZR HSN   SN;          % 13d.  If G is ZR   AND  H  is SN      THEN  P is ZR
GZR HLN   LN;          % 13e.  If G is ZR   AND  H  is LN      THEN  P is SN

GSN HLP   SP;          % 14a.  If G is SN   AND  H  is LP      THEN  P is ZR
GSN HSP   ZR;          % 14b.  If G is SN   AND  H  is SP      THEN  P is SN
GSN HZR   SN;          % 14c.  If G is SN   AND  H  is ZR      THEN  P is SN
GSN HSN   SN;          % 14d.  If G is SN   AND  H  is SN      THEN  P is SN
GSN HLN   LN;          % 14e.  If G is SN   AND  H  is LN      THEN  P is SN

GLN HLP   LN;          % 15a.  If G is LN   AND  H  is LP      THEN  P is LN
GLN HSP   LN;          % 15b.  If G is LN   AND  H  is SP      THEN  P is LN
GLN HZR   LN;          % 15c.  If G is LN   AND  H  is ZR      THEN  P is LN
GLN HSN   LN;          % 15d.  If G is LN   AND  H  is SN      THEN  P is LN
GLN HLN   LN ];        % 15e.  If G is LN   AND  H  is LN      THEN  P is LN

% ***  fuzzy logic to combine azimuth, hca and range info for t_now bank  ***

% append column for logical AND of now_azi_hca and rng_cmbank for bank angle
now_bank(:,4) = min(now_bank(:,1),now_bank(:,2));
now_bank(:,5) = now_bank(:,3).*now_bank(:,4);
now_bank = sum(now_bank(:,5))/sum(now_bank(:,4));
```

```
%  ************************************************************************
%         t_next logic for computing bank to next set point              *
%     based on future radar gimbal azimuth and CM to CMCA HCA            *
%  ************************************************************************

% Convert scaled next_azi and next_hca bank values to fuzzy set levels
ef_bank = [
ELP FLP   SN;          % 16a. If E is LP   AND   F is LP      THEN  Q is LP
ELP FSP   SN;          % 16b. If E is LP   AND   F is SP      THEN  Q is LP
ELP FZR   SP;          % 16c. If E is LP   AND   F is ZR      THEN  Q is SP
ELP FSN   SP;          % 16d. If E is LP   AND   F is SN      THEN  Q is ZR
ELP FLN   SP;          % 16e. If E is LP   AND   F is LN      THEN  Q is ZR

ESP FLP   SN;          % 17a. If E is SP   AND   F is LP      THEN  Q is LP
ESP FSP   ZR;          % 17b. If E is SP   AND   F is SP      THEN  Q is SP
ESP FZR   ZR;          % 17c. If E is SP   AND   F is ZR      THEN  Q is ZR
ESP FSN   SP;          % 17d. If E is SP   AND   F is SN      THEN  Q is ZR
ESP FLN   SP;          % 17e. If E is SP   AND   F is LN      THEN  Q is SN

EZR FLP   SN;          % 18a. If E is ZR   AND   F is LP      THEN  Q is LP
EZR FSP   SN;          % 18b. If E is ZR   AND   F is SP      THEN  Q is SP
EZR FZR   ZR;          % 18c. If E is ZR   AND   F is ZR      THEN  Q is ZR
EZR FSN   SP;          % 18d. If E is ZR   AND   F is SN      THEN  Q is SN
EZR FLN   SP;          % 18e. If E is ZR   AND   F is LN      THEN  Q is LN

ESN FLP   SN;          % 19a. If E is SN   AND   F is LP      THEN  Q is SP
ESN FSP   SN;          % 19b. If E is SN   AND   F is SP      THEN  Q is ZR
ESN FZR   ZR;          % 19c. If E is SN   AND   F is ZR      THEN  Q is ZR
ESN FSN   ZR;          % 19d. If E is SN   AND   F is SN      THEN  Q is SN
ESN FLN   SP;          % 19e. If E is SN   AND   F is LN      THEN  Q is LN

ELN FLP   SN;          % 20a. If E is LN   AND   F is LP      THEN  Q is ZR
ELN FSP   SN;          % 20b. If E is LN   AND   F is SP      THEN  Q is ZR
ELN FZR   SN;          % 20c. If E is LN   AND   F is ZR      THEN  Q is SN
ELN FSN   SP;          % 20d. If E is LN   AND   F is SN      THEN  Q is LN
ELN FLN   SP ];        % 20e. If E is LN   AND   F is LN      THEN  Q is LN

% append column for logical AND of next_azi and next_hca for bank angle
ef_bank(:,4) = min(ef_bank(:,1),ef_bank(:,2));
ef_bank(:,5) = ef_bank(:,3).*ef_bank(:,4);
next_bank = sum(ef_bank(:,5))/sum(ef_bank(:,4));
```

```
% ************************************************************************
%         logic to combine present and future bank commands              *
% ************************************************************************

% Convert scaled present & future cmd'd bank values to fuzzy set levels
[PLP,PSP,PZR,PSN,PLN] = s2fuz(now_bank);      % fuzzy bank cmd at t_now
[QLP,QSP,QZR,QSN,QLN] = s2fuz(next_bank);     % fuzzy bank cmd at t_next

net_bank = [
PLP QLP   LP;      % 21a.  If P is LP   AND   Q is LP      THEN  R is LP
PLP QSP   LP;      % 21b.  If P is LP   AND   Q is SP      THEN  R is LP
PLP QZR   LP;      % 21c.  If P is LP   AND   Q is ZR      THEN  R is LP
PLP QSN   LP;      % 21d.  If P is LP   AND   Q is SN      THEN  R is LP
PLP QLN   LP;      % 21e.  If P is LP   AND   Q is LN      THEN  R is LP

PSP QLP   LP;      % 22a.  If P is SP   AND   Q is LP      THEN  R is LP
PSP QSP   SP;      % 22b.  If P is SP   AND   Q is SP      THEN  R is SP
PSP QZR   ZR;      % 22c.  If P is SP   AND   Q is ZR      THEN  R is ZR
PSP QSN   ZR;      % 22d.  If P is SP   AND   Q is SN      THEN  R is SN
PSP QLN   SN;      % 22e.  If P is SP   AND   Q is LN      THEN  R is SN

PZR QLP   SP;      % 23a.  If P is ZR   AND   Q is LP      THEN  R is SP
PZR QSP   SP;      % 23b.  If P is ZR   AND   Q is SP      THEN  R is SP
PZR QZR   ZR;      % 23c.  If P is ZR   AND   Q is ZR      THEN  R is ZR
PZR QSN   SN;      % 23d.  If P is ZR   AND   Q is SN      THEN  R is SN
PZR QLN   SN;      % 23e.  If P is ZR   AND   Q is LN      THEN  R is SN

PSN QLP   SP;      % 24a.  If P is SN   AND   Q is LP      THEN  R is SP
PSN QSP   ZR;      % 24b.  If P is SN   AND   Q is SP      THEN  R is SP
PSN QZR   ZR;      % 24c.  If P is SN   AND   Q is ZR      THEN  R is ZR
PSN QSN   SN;      % 24d.  If P is SN   AND   Q is SN      THEN  R is SN
PSN QLN   LN;      % 24e.  If P is SN   AND   Q is LN      THEN  R is LN

PLN QLP   LN;      % 25a.  If P is LN   AND   Q is LP      THEN  R is LN
PLN QSP   LN;      % 25b.  If P is LN   AND   Q is SP      THEN  R is LN
PLN QZR   LN;      % 25c.  If P is LN   AND   Q is ZR      THEN  R is LN
PLN QSN   LN;      % 25d.  If P is LN   AND   Q is SN      THEN  R is LN
PLN QLN   LN ];    % 25e.  If P is LN   AND   Q is LN      THEN  R is LN

% append column for logical AND of now_bank and next_bank
net_bank(:,4) = min(net_bank(:,1),net_bank(:,2));
net_bank(:,5) = net_bank(:,3).*net_bank(:,4);
net_bank_mean = sum(net_bank(:,5))/sum(net_bank(:,4));

% *****  Convert fuzzy set commanded bank value to hard value  *****

cmd_bank = net_bank_mean / 5 * accel_limits(6);

return;
```

```
function  [SLP,SSP,SZR,SSN,SLN] = s2fuz(scaled_level)
%                                                                    S2FUZ.M
%                                                                    8 Feb 93
%                                                                    Randy Nelson
% This file takes a scaled level and determines membership
%  level in all linguistic sets
%
%  Input: scaled level [-5,5] (extra values of ±99 on each end
%    preclude out-of-range problems)
%
%  Output is a 5 x 1 vector of set membership levels
%   All non-zero values of "set_members" have membership.
%   In order "scaled large positve," "scaled small positive,"
%     "zero," "small negative," and "large negative"
%

%  *****  5 Linguistic sets and 33 scaled levels [-5,5]  *****
%   define sets as trapezoids

index = [-99 -5:5 99];                  % scaled "universe of discourse"

Linguistic_set = [
%  LN   SN   ZR   SP   LP    Scaled level
   1    0    0    0    0;
   1    0    0    0    0;    % -5
   1    0    0    0    0;    % -4
   0    1    0    0    0;    % -3
   0    1    0    0    0;    % -2
   0    0    1    0    0;    % -1
   0    0    1    0    0;    %  0
   0    0    1    0    0;    %  1
   0    0    0    1    0;    %  2
   0    0    0    1    0;    %  3
   0    0    0    0    1;    %  4
   0    0    0    0    1;    %  5
   0    0    0    0    1 ];

  tab = [index' Linguistic_set];

out = table1(tab,scaled_level);
SLN = out(1);
SSN = out(2);
SZR = out(3);
SSP = out(4);
SLP = out(5);

return;
```

```
function Lout = sat_xyz(u)
%                                                                    SAT_XYZ.M
%                                                                    9 Jan 93
%                                                                    Randy Nelson
% Input: a 7 x 1 column vector
%   The four inertial controls (accelerations)
%   (phi acceleration is not used),
%   and the euler angles (psi, theta and phi),
%
%  Uses accel_limits from "runsim.m," a 1 x 8 vector.
%
% Output: a 3 x 1 vector:
%   The body x, y and z axis accelerations, limited.

psi   = u(5);
theta = u(6);
phi   = u(7);

zero_tol = 1.0e-4;          % used towards end of file when scaling accel
if 0   % not implemented, but available if needed
  if abs(psi) < zero_tol
    psi = 0;
   end; % if abs(psi) < zero_tol
  if abs(theta) < zero_tol
    theta = 0;
   end; % if abs(theta) < zero_tol
  if abs(phi) < zero_tol
    phi = 0;
   end; % if abs(phi) < zero_tol
 end; % if 0

%  ******************    Compute transformation matrix D  *************
D = zeros(3,3);                              % speeds up memory allocation a bit
D(1,1) = cos(theta)*cos(psi);
D(1,2) = cos(theta)*sin(psi);
D(1,3) = -sin(theta);
D(2,1) = sin(phi)*sin(theta)*cos(psi) - cos(phi)*sin(psi);
D(2,2) = sin(phi)*sin(theta)*sin(psi) + cos(phi)*cos(psi);
D(2,3) = sin(phi)*cos(theta);
D(3,1) = cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi);
D(3,2) = cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi);
D(3,3) = cos(phi)*cos(theta);

Lb = D * u(1:3);                             % convert inertial accel's to body
Gb = D*[0 0 gravity]';                       % convert inertial gravity to body
Lb1 = Lb;

%  **********  limit X accel  **********
Lower = max([accel_limits(1) (vcmca_min-vcmca)]);
Upper = min([accel_limits(2) (vcmca_max-vcmca)]);

if (Lb(1) > Upper)                            % X limited by throttle
  Lb(1) = Upper;
 elseif (Lb(1) < Lower)
  Lb(1) = Lower;
 end; % if (Lb(1) > Upper)
if (Upper < Lower)
  Lb(1) = min([Lower Upper]);
 end; % if (Upper < Lower)
Lbout(1) = Gb(1) + Lb(1);                     % X further affected by gravity
```

```
%  **********  limit Z accel  **********
if (phi*Lb1(2) > 0)                          % rescale commanded control
  Lb(3) = Lb(3)*Gb(2)/Lb(2);
 elseif (phi*Lb1(2) < 0)                     % banked in wrong direction
  Lb(3) = (-gravity + u(3))/cos(phi) + Gb(3);
 end; % if (phi*Lb1(2) > 0)
Lb(3) = -Gb(3) + Lb(3);                      % Z due to gravity and backpressure
if (Lb(3) < -accel_limits(5))                % don't over-g
  Lb(3) = -accel_limits(5);
 elseif (Lb(3) > -accel_limits(4))           % don't under-g
  Lb(3) = -accel_limits(4);
 end; % if (Lb(3) < -accel_limits(5))

%  **********  limit Y accel  **********
Lb(2) = 0;

%  can go at top of function w/return, do it here to get debug output
if (abs(Lb1(2)) < zero_tol) & (abs(Lb1(3)) < zero_tol)
  g_s = 1/cos(phi);
  Lout = D'*[Lbout(1) Lb(2) -gravity*g_s]';
%disp('here1')
 else
  Lout = D'*[Lbout(1)  Lb(2)  Lb(3)]';
  g_s = -Lb(3)/gravity;
%disp('here2')
 end; % if (Lb1(2) == 0) & (Lb1(3) == 0)
Lout(3) = Lout(3) + gravity;

return;
```

```
function  setp = set_pt(u)
%                                                                    SET_PT.M
%                                                                    24 Jan 93
%                                                                    Randy Nelson
%
% Uses fuzzy logic to compute set point in CMMCA body axis
%  Fuzzy logic provides a recommended CMMCA bank angle and speed change
%  This info is used to determine where the CMMCA would be at the
%   next time interval assuming it was at the desired bank throughout
%   and the airspeed was changed (once) by the desired amount
%
%  Inputs: a 23 x 1 vector of CM & CMMCA states,
%    euler angles, radar range and gimbal angles,
%    and current time (t_now)
%   Uses: CMMCA desired bank angle and speed change from "log_tab.m,"
%     fuzzy_time (ftut,t_reset,t_ahead) from "time_fuz.m,"
%     CM_next (for next_azi, next_hca)
%
%  Outputs: CMMCA set point in inertial space (8 x 1 vector)
%

%  *****  Attach names to inputs  *****
cm_state    = u(1:8);
cmmca_state = u(9:16);
psi         = u(17);
theta       = u(18);
phi         = u(19);
rng         = u(20);
gim_psi     = u(21);
gim_theta   = u(22);
t_now       = u(23);

if isempty(sim_start_time)            % use Matlab's clock to figure
  sim_start_time = clock;             % elapsed time of simulation
 end
if t_now > tfinal - .25
  sim_run_tim = etime(clock,sim_start_time)
 end


% ********  A COUPLE OF CONTINGENCIES BEFORE FINDING NEXT SET POINT  *********

%  *****  skip fuzzy logic and revert to "refpos.M"  *****
if (fuz_enable == -1)
  setp = refpos([cm_state; cmmca_state]);
  return;
 end; % if (fuz_enable == -1)

%  *****  don't recompute set point if time < reset time  *****
if (t_now < t_reset)            % t_reset = 0 initially, so this is skipped
  setp = [setpoint(1:3); cmmca_state(4:8)];
  setpoint = setp;              % set the global variable to just computed value
  return;
 end; % if (t_now < t_reset)

% *****   recompute set point if time >= reset time  *****
%                    or if time = tstart
%  ***  get new set of fuzzy times from matrix  ***
time_index = 1+round(t_now/start_step);     % index into row of fuzzy_time
delta_ftut = fuzzy_time(time_index,1);
ftut = ftut + delta_ftut;
t_reset = t_reset + fuzzy_time(time_index,2);
```

```
% get CM bank, heading and heading crossing angle
t_round = 1+round(t_now);                          % index into row of cm_hpb
cm_bank = cm_hpb(t_round,4);
cm_hdg = cm_hpb(t_round,1);                        % from matrix of values
delta_hdg = rem(2*pi + cm_hdg - psi,2*pi);
if delta_hdg > pi                                  % in range [-180,180]
  delta_hdg = delta_hdg - 2 * pi;
 end; % if delta_hdg > pi

% get future cm heading and heading crossing angle
[next_hdg,next_azi] = ...
                    cm_next(t_now + t_ahead,cmmca_state,psi);

% future HCA based on future CM heading and present CMMCA heading
next_hca = rem(2*pi + next_hdg - psi,2*pi);
if next_hca > pi                                   % in range [-180,180]
  next_hca = next_hca - 2 * pi;
 end; % if next_hca > pi

%  ***** compute the set point in the body axes  *****
[cmmca_bank,cmmca_speed] = ...
                   log_bank(gim_psi,delta_hdg,rng,next_azi,next_hca,cm_bank);

vel = norm(cmmca_state(5:7));          % Compute CMMCA speed
vel = vel + cmmca_speed;               %  and incorporate change

if cmmca_bank == 0
  body_xy = [vel*delta_ftut  0];
 else
  k1 = vel^2/(gravity*tan(cmmca_bank));
  k2 = (gravity*delta_ftut*tan(cmmca_bank))/vel;
  body_xy = k1 * [sin(k2)  1-cos(k2)];
 end; % if cmmca_bank == 0

%  ***** transform into the (body centered) inertial axes *****
inert = body2inert([body_xy 0 psi 0 0]);    % ref'd to zero pitch & bank

%  ***** convert to the (true origin) inertial axes *****
xyz = inert + cmmca_state(1:3);

setp = [xyz; cmmca_state(4:8)];
setpoint = setp;                % set the global variable to just computed value

return;
```

Bibliography

1. Bryson, Arthur E. and Ho, Yu-Chi. *Applied Optimal Control: Optimization, Estimation, and Control*. Waltham MA: Blaisdell Publishing Company, 1975.

2. Caughlin, Col Donald J., Associate Dean. Computer Software. Air Force Institute of Technology, Wright Patterson AFB OH, December 1992.

3. Chiu, Stephen, et al. "Fuzzy Logic for Control of Roll and Moment for a Flexible Wing Aircraft," *Proceedings of Fifth IEEE International Symposium on Intelligent Control*. 42-48 Philadelphia:IEEE Press, 1991.

4. CONTROL SYSTEM Toolbox. Version 3.03, SUN and IBM. Computer Software. The MathWorks, Inc. Natick MA, 1990.

5. Cox, Earl. "Fuzzy Fundamentals," *IEEE Spectrum* 58-61 (October 1992).

6. Garton, Capt Antoine M. *Using Simulation to Determine a Strategy for Positively Tracking a Cruise Missile by CMMCA*. MS Thesis, AFIT/GST/ENS/90M-6. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, March 1990. (AD-A220525)

7. Hachman, Capt Andrew C. *Developing CMMCA Flight Profiles for Cruise Missile Tracking*. MS Thesis, AFIT/GOR/ENS/92M-13. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, March 1992. (AD-A248183)

8. Heavner, Capt Leonard G. *An Investigation into the Application of Dynamic Programming to Determine the Optimal Controls for Tracking a Cruise Missile Maneuver by CMMCA*. MS Thesis, AFIT/GST/ENS/89M-6. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, March 1989. (AD-A205970)

9. Kailath, Thomas. *Linear Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1980.

10. Lewis, Frank L. *Optimal Control*. New York: John Wiley & Sons, 1986.

11. Li, Y.F. and Lau, C.C. "Development of Fuzzy Algorithms for Servo Systems," *IEEE Control Systems Magazine*, 65-71 (April 1989).

12. Maybeck, Peter S. *Stochastic models, Estimation, and Control, Volume 1*. New York: Academic Press, 1979.

13. Menon, P.K.A. "Short-Range Nonlinear Feedback Strategies for Aircraft Pursuit-Evasion," *Journal of Guidance, Control and Dynamics*, 12:27-32 (Jan-Feb 1989).

14. Menon, P.K.A. and Duke, E.L. "Time-Optimal Aircraft Pursuit Evasion with a Weapon Envelope Constraint," *Journal of Guidance, Control and Dynamics*, 15:448-456 (March-April 1992).

15. PC-MATLAB. Version 3.5a, IBM. Computer Software. The MathWorks, Inc. Natick MA, 1991.

16. PRO-MATLAB. Version 3.45, Sun SPARCstation 2. Computer Software. The MathWorks, Inc. Natick MA, 1989.

17. SIMULINK. Version 1.2.1, SUN and IBM. Computer Software. The MathWorks, Inc. Natick MA, 1992.

18. SPLINE Toolbox. Version 1.02, SUN and IBM. Computer Software. The MathWorks, Inc. Natick MA, 1990.

19. Winston, Wayne L. *Mathematical Programming: Applications and Algorithms*. Boston: PWS-Kent Publishing Company, 1991.

Vita

Major Randy E. Nelson was born in Seattle, Washington on 5 May 1958. He graduated from Queen Anne High School in Seattle in 1976. He attended the U.S. Air Force Academy, graduating in May, 1980 with Bachelor of Science degrees in Chemistry and Physics, and a regular commission as a Second Lieutenant in the USAF. He attended Undergraduate Pilot Training in Columbus, Mississippi, and upon receiving his pilot wings, attended follow on F-4 training in Tampa, Florida. He was successively assigned to the 52nd Tactical Fighter Wing in the Federal Republic of Germany from 1982 until 1985 and the 3rd Tactical Fighter Wing in the Republic of the Philippines from 1985 until 1988. He was then returned to Undergraduate Pilot Training as an T-37 Instructor Pilot at Laughlin AFB, Texas. While there he held the positions of Flight Commander, Chief of Check Section and Assistant Operations Officer for the 85th and then the 84th Flying Training Squadron. In 1991 he was selected for the School of Engineering, Air Force Institute of Technology, Wright Patterson AFB, Ohio. Following graduation with a Master of Science in Operations Research (Strategic and Tactical Analysis) in March, 1993, he will be assigned to Studies and Analysis, The Pentagon, Washington D.C.

Permanent Address: 3009 Silver Lake Court
Adelphi, MD 20783

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1993	Master's Thesis

4. TITLE AND SUBTITLE

A FUZZY LOGIC OPTIMAL CONTROL LAW SOLUTION TO THE CMMCA TRACKING PROBLEM

5. FUNDING NUMBERS

6. AUTHOR(S)

Randy E. Nelson, Major, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GST/ENS/93M-10

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N\A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Air Force uses the C-18 Cruise Missile Mission Control Aircraft (CMMCA) to radar track cruise missiles (CM) during test flights. Because of the complexity of the CM flight profiles, maintaining radar coverage at all times is very difficult. This thesis attempted to apply optimal control theory to construct a simulation providing 100% radar coverage. The simulation was divided into ten second intervals, and fuzzy logic was used at the start of each interval to determine the set point, i.e., that point in space where the CMMCA should be in ten seconds. The set point calculation's fuzzy logic balanced CMMCA maneuvering based on present and future CM positions. Three different future times were used: 60, 90 and 120 seconds ahead, and the performance for each time was compared. The simulation was performed on an IBM compatible PC with Matlab and Simulink (both by The MathWorks). The final form of the fuzzy logic provided varying radar coverage at each look ahead time for a complex CM flight path (CM in 20 degrees of bank) 1850 seconds long. At 120 seconds look ahead time, the coverage was 100%. When the same profile was performe with the CM in 30 degrees of bank, coverage was degraded, and 60 seconds look ahead performed best.

14. SUBJECT TERMS

CMMCA, Cruise Missile Tracking, Fuzzy Logic, Optimal Control, Flight Simulation, Computer Simulation, Guidance

15. NUMBER OF PAGES
146

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL